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

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 Demo Program: The Depth Shadow Map

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

The rendering is done into a texture and only renders the depth, not any colors. (Normally, we would render both, but in this case, we only care about the depth.)

In this grayscale depth 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:
`glGenFramebuffers( 1, &DepthFramebuffer );`
`glGenTextures( 1, &DepthTexture );`

2. Create a texture that will be the framebuffer's depth buffer
`glBindTexture(GL_TEXTURE_2D, DepthTexture );`

3. Bind the texture to the current framebuffer as a depth buffer:
`glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT, SHADOW_WIDTH, SHADOW_HEIGHT, 0, GL_FLOAT, NULL);`

4. 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);`

5. Bind the framebuffer:
`glBindFramebuffer(GL_FRAMEBUFFER, DepthFramebuffer);`

OpenGL Demo Program: Rendering into the Depth Shadow Map

1. Render to the depth shadow map:
`glReadBuffer(GL_NONE);`
`glDrawBuffer(GL_NONE);`

2. Force OpenGL to accept a framebuffer that doesn't have a color buffer in it:
`glBindFramebuffer(GL_FRAMEBUFFER, 0);`

OpenGL Demo Program: Rendering using the Depth Shadow Map

1. Render with shadows:
   - Set the light position and view matrices:
   - Bind the framebuffer and depth texture:
   - Set uniforms:
   - Render the scene:

2. Render without shadows:
   - Set the light position and view matrices:
   - Bind the framebuffer and depth texture:
   - Set uniforms:
   - Render the scene:

OpenGL Demo Program: Creating the Off-screen Depth Shadow Map

1. Create the OpenGL demo program:
2. Initialize the graphics environment:
3. Create the depth shadow map:
4. Render the scene with shadows:
5. Render the scene without shadows:

OpenGL Demo Program: Rendering using the Depth Shadow Map

1. Render the scene with shadows:
2. Render the scene without shadows:

OpenGL Demo Program: Setting uniforms:

1. Set the light position and view matrices:
2. Bind the framebuffer and depth texture:
3. Set uniforms:
4. Render the scene:

OpenGL Demo Program: Displaying the Depth Shadow Map

1. Display the scene with shadows:
2. Display the scene without shadows:

OpenGL Demo Program: Setting uniforms:

1. Set the light position and view matrices:
2. Bind the framebuffer and depth texture:
3. Set uniforms:
4. Render the scene:

OpenGL Demo Program: Displaying the Depth Shadow Map

1. Display the scene with shadows:
2. Display the scene without shadows:

OpenGL Demo Program: Setting uniforms:

1. Set the light position and view matrices:
2. Bind the framebuffer and depth texture:
3. Set uniforms:
4. Render the scene:
Uniform vec3 uColor;

Uniform sampler2D uShadowMap;

In vec4 vFragPosLightSpace;

In vec3 vNs;

In vec3 vLs;

In vec3 vEs;

const float BIAS = 0.01;

const vec3 SPECULAR_COLOR = vec3(1., 1., 1.);

const float SHININESS = 8;

const float KA = 0.20;

const float KD = 0.60;

const float KS = (1.-KA-KD);

bool IsInShadow( vec4 fragPosLightSpace )
{
  // have to manually do homogenous division to make light space position in range of -1 to 1:
  vec3 projection = vFragPosLightSpace.xyz / vFragPosLightSpace.w;
  // then make it from 0 to 1:
  projection = 0.5*projection + 0.5;
  // Get closest depth from light's perspective
  float closestDepth = texture( uShadowMap, projection.xy ).r;
  // Get current depth:
  float currentDepth = projection.z;
  bool isInShadow = (currentDepth - BIAS) > closestDepth;
  return isInShadow;
}

Uniform mat4 uModel;

Uniform mat4 uView;

Uniform mat4 uProj;

Out vec2 vST;

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

How Did the Demo Program Render the 2D Shadow Map?

DisplayShadowMap.vert

uniform mat4 uModel;

uniform mat4 uView;

uniform mat4 uProj;

out vec2 vST;

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

DisplayShadowMap.frag

float gray = texture( uShadowMap, vST ).r;

gl_FragColor = vec4( gray, gray, gray, 1. );

OpenGL Demo Program: Rendering using the Depth Shadow Map

OpenGL Demo Program: Rendering into the Depth Shadow Map

OpenGL Demo Program: Rendering using the Depth Shadow Map

How Did the Demo Program Render that 2D Shadow Map?

Note that this is not a required part of a shadows program. I just did that to show you what it would look like.

DisplayShadowMap.vert

uniform mat4 uModel;

uniform mat4 uView;

uniform mat4 uProj;

out vec2 vST;

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

DisplayShadowMap.frag

float gray = texture( uShadowMap, vST ).r;

gl_FragColor = vec4( gray, gray, gray, 1. );

OpenGL Demo Program: Rendering using the Depth Shadow Map

OpenGL Demo Program: Rendering using the Depth Shadow Map

OpenGL Demo Program: Rendering into the Depth Shadow Map

OpenGL Demo Program: Rendering using the Depth Shadow Map

OpenGL Demo Program: Rendering using the Depth Shadow Map