Casting Shadows in OpenGL
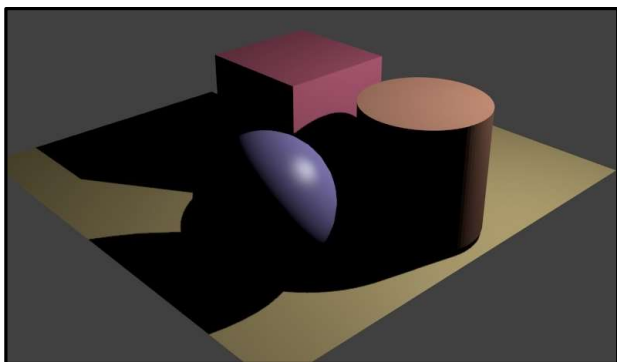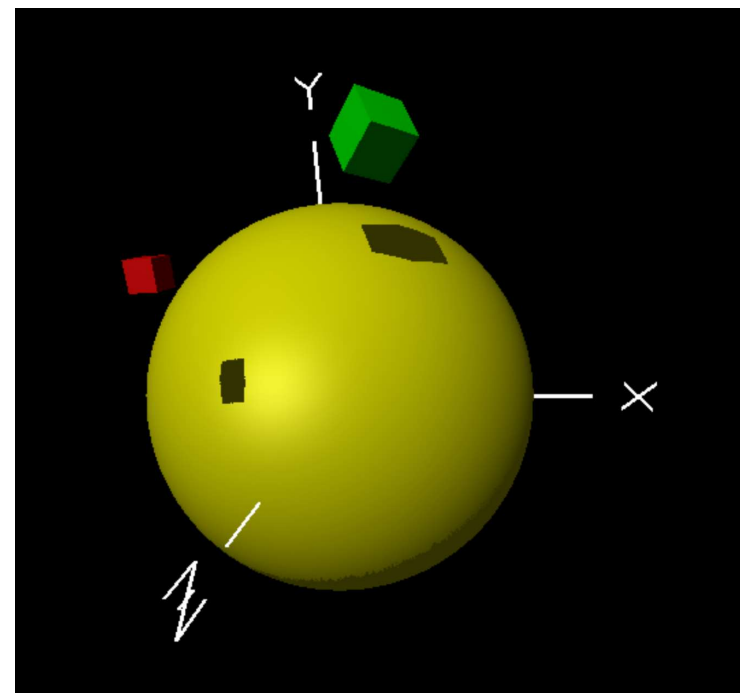
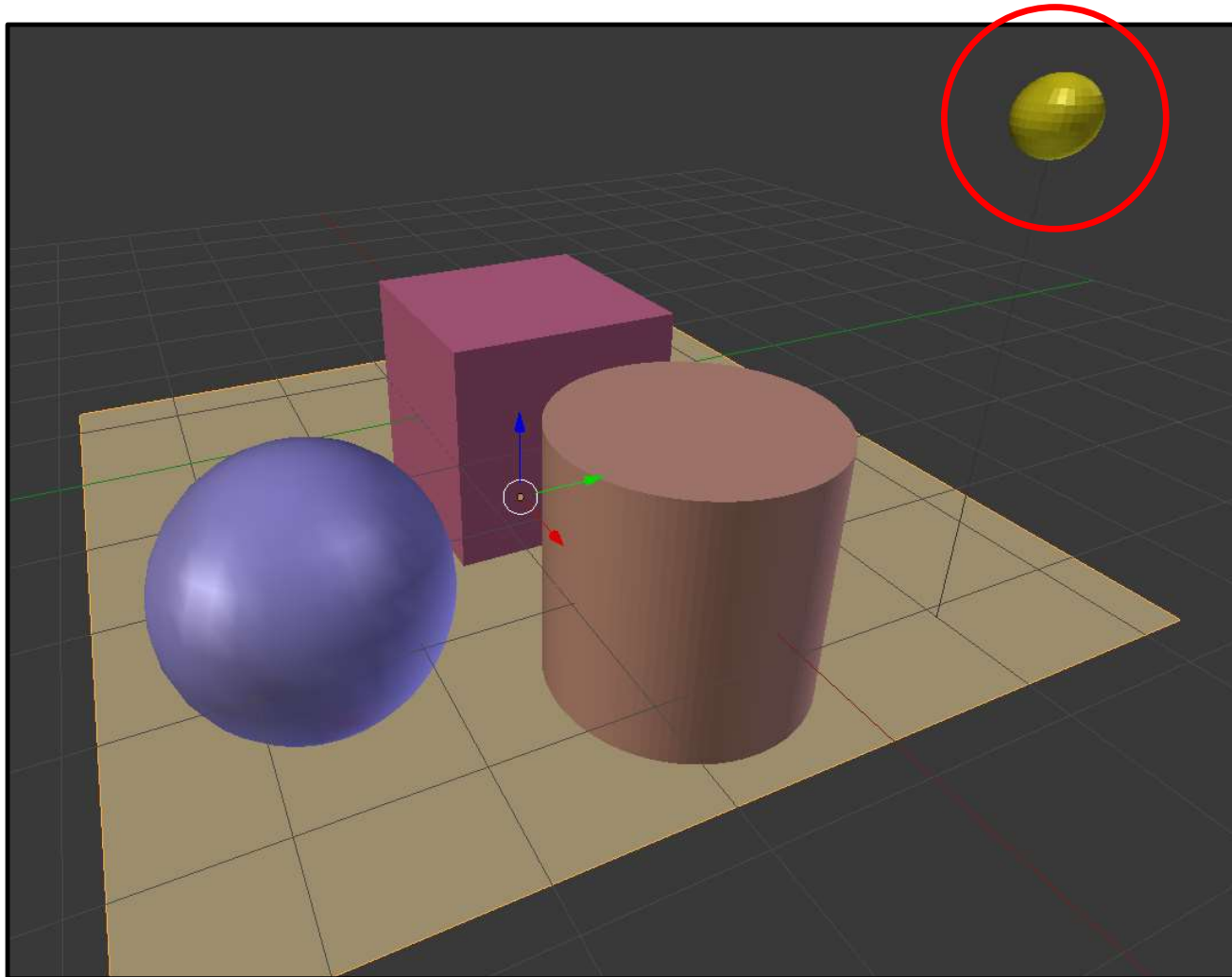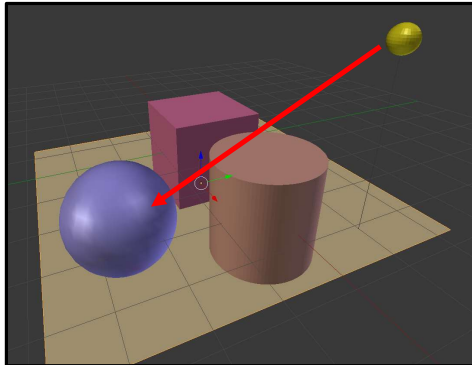**Mike Bailey**

mjb@cs.oregonstate.edu

Oregon State
University
Computer Graphics

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
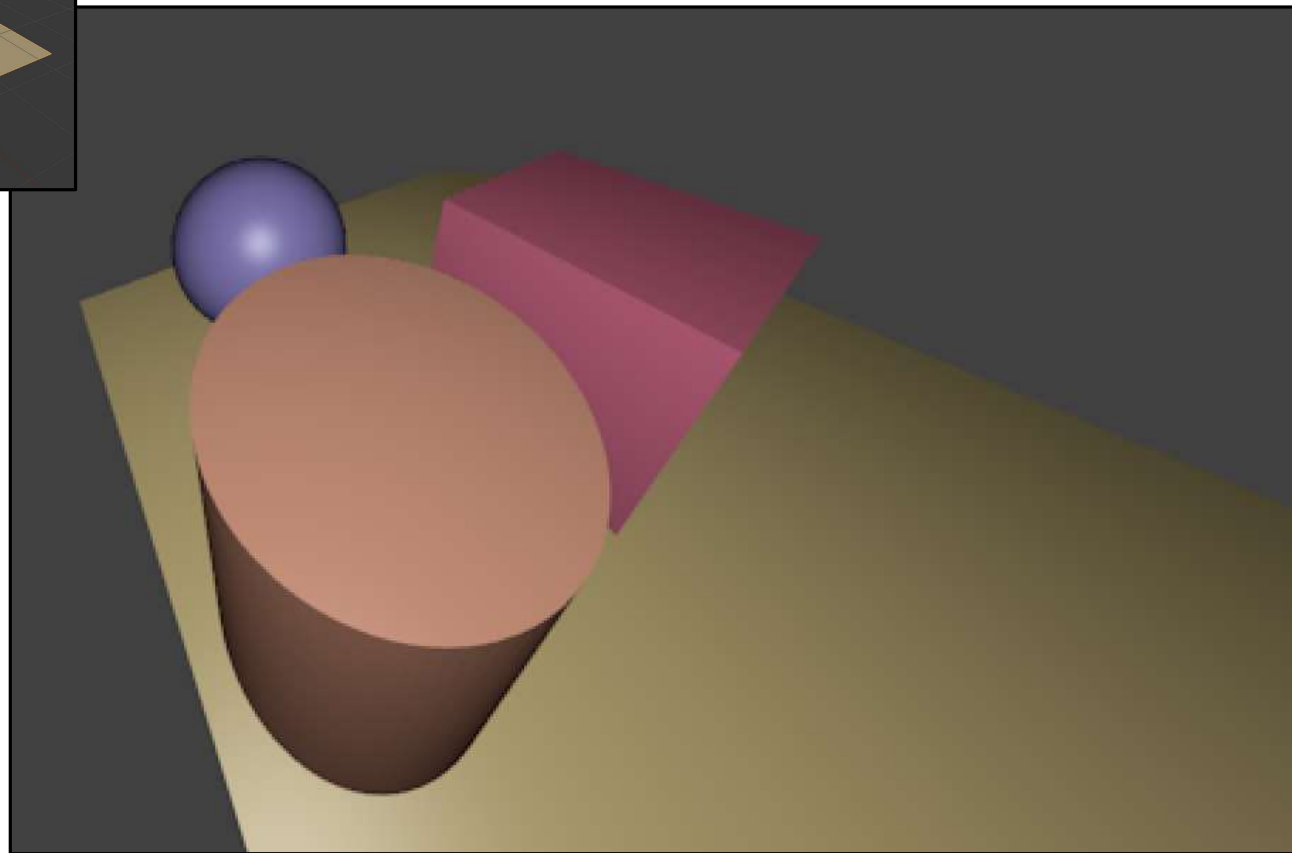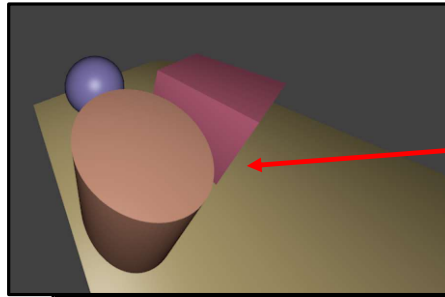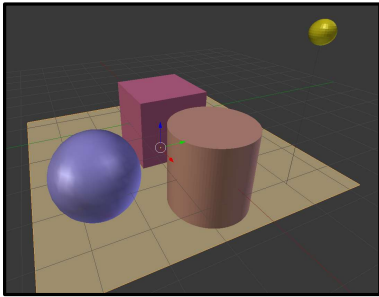
2. Generate a depth view from the light source
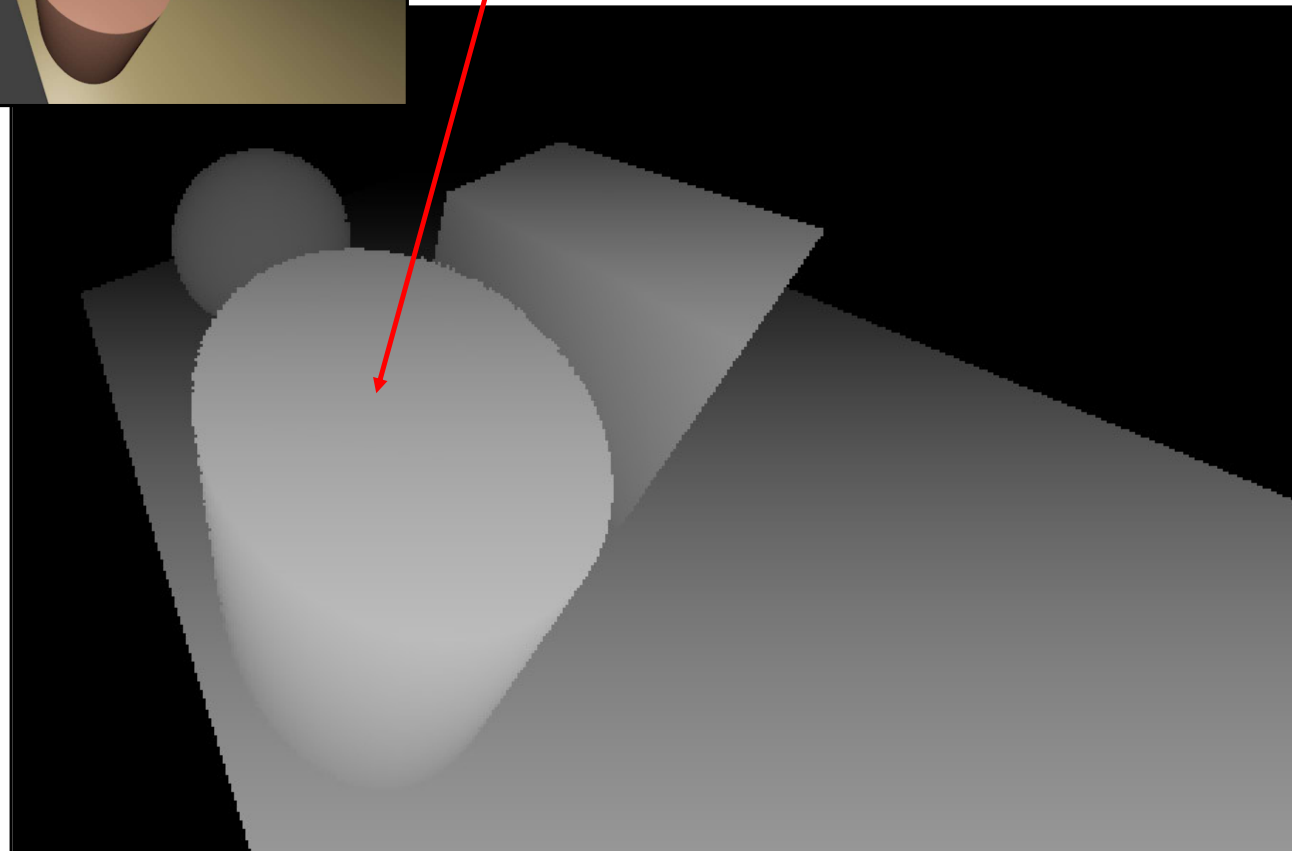
Second, Render the Scene as Normal, but Consult the Depth Map to Decide where Lighting Applies

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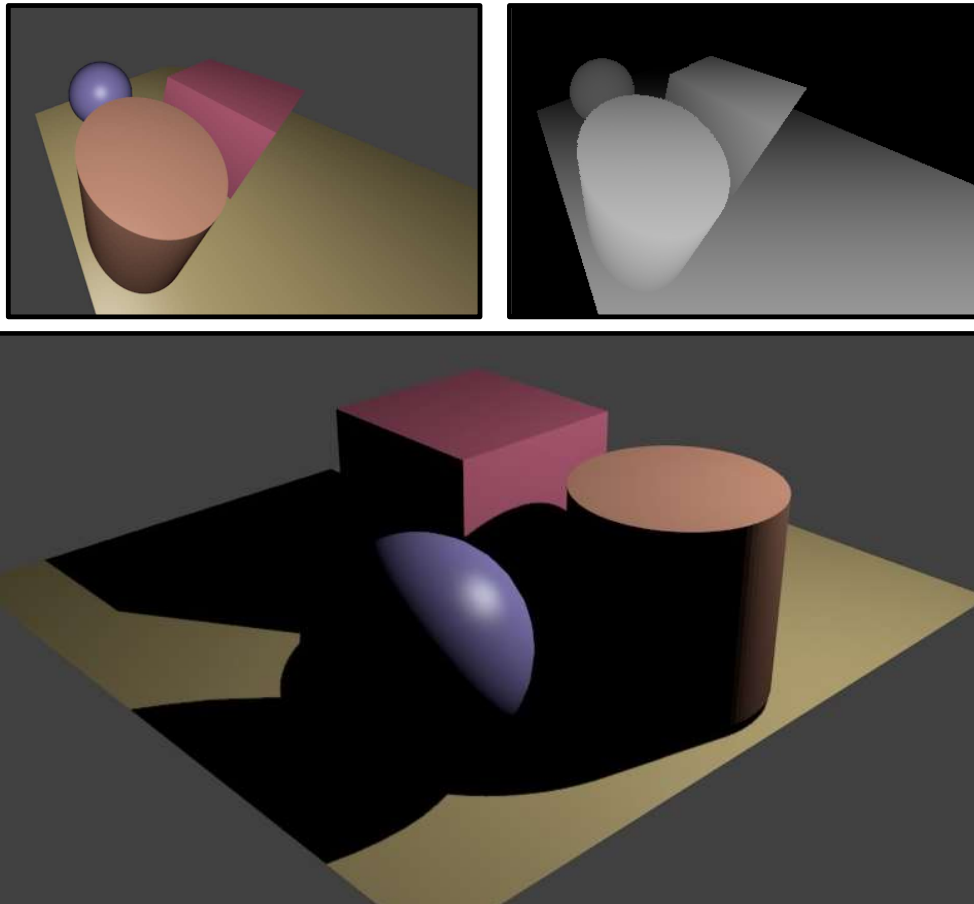
The depth shadow map is created from the point of view of the light source.

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.
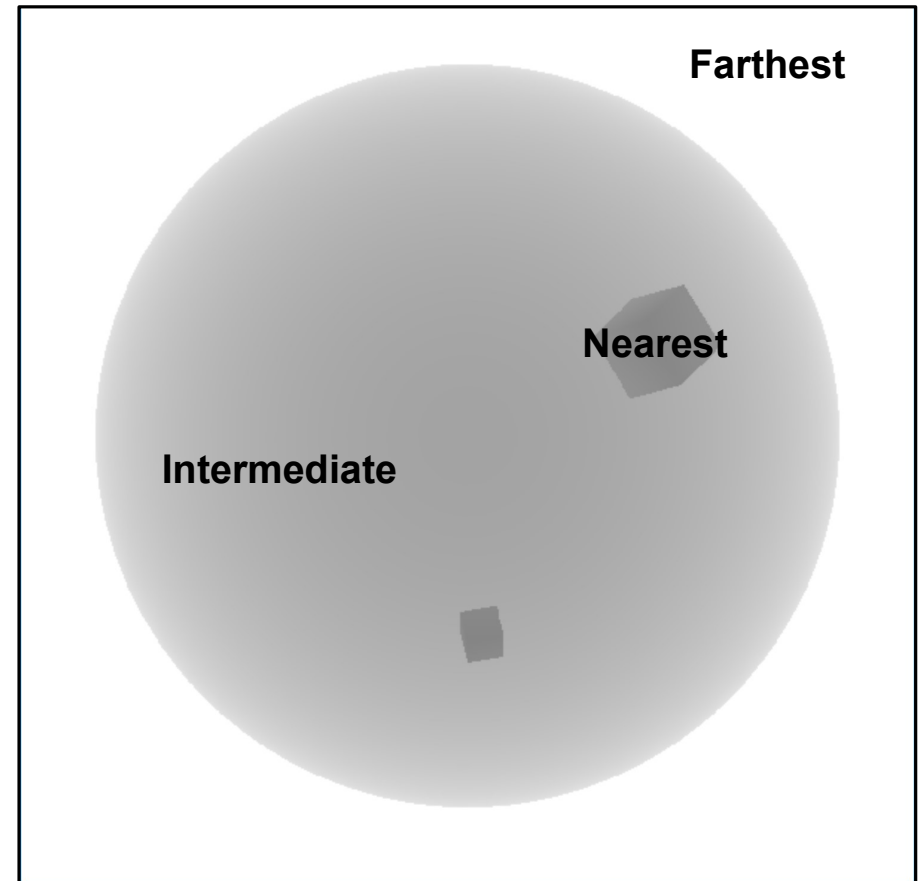
Oregon State
University
Computer Graphics

In shadows.cpp:
InitGraphics( )

```
// create a framebuffer object and a depth texture object:
glGenFramebuffers(1, &DepthFramebuffer );
glGenTextures(     1, &DepthTexture );

//Create a texture that will be the framebuffer's depth buffer
glBindTexture(GL_TEXTURE_2D, DepthTexture );
glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT, SHADOW_WIDTH, SHADOW_HEIGHT,
             0, GL_DEPTH_COMPONENT, GL_FLOAT, NULL);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);

// attach texture to framebuffer as depth buffer:
glBindFramebuffer(GL_FRAMEBUFFER, DepthFramebuffer);
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT, GL_TEXTURE_2D, DepthTexture, 0);

// force opengl to accept a framebuffer that doesn't have a color buffer in it:
glDrawBuffer(GL_NONE);
glReadBuffer(GL_NONE);
glBindFramebuffer(GL_FRAMEBUFFER, 0);
```

Farthest

Nearest

Intermediate

University
Computer Graphics

OpenGL Demo Program: Rendering into the Depth Shadow Map

```
//first pass, render from light's perspective, store depth of scene in texture
glBindFramebuffer(GL_FRAMEBUFFER, DepthFramebuffer);
glClear(GL_DEPTH_BUFFER_BIT);
glDrawBuffer(GL_NONE);
glReadBuffer(GL_NONE);
glEnable(GL_DEPTH_TEST);
glShadeModel(GL_FLAT);
glDisable(GL_NORMALIZE);

// these matrices are the equivalent of projection and view matrices
glm::mat4 lightProjection = glm::ortho(-10.0f, 10.0f, -10.0f, 10.0f, 1.f, 20.f);
glm::vec3 lightPos(LightX, LightY, LightZ);
glm::mat4 lightView = glm::lookAt(lightPos, glm::vec3(0., 0., 0.), glm::vec3(0., 1., 0.));

//this matrix is the transformation matrix that the vertex shader will use instead of glModelViewProjectionMatrix:
glm::mat4 lightSpaceMatrix = lightProjection * lightView;

glViewport(0, 0, SHADOW_WIDTH, SHADOW_HEIGHT);

GetDepth->Use( );
GetDepth->SetUniformVariable((char*)"uLightSpaceMatrix", lightSpaceMatrix);
glm::vec3 color = glm::vec3(0., 1., 1.);
GetDepth->SetUniformVariable((char*)"uColor", color);
DisplayOneScene(GetDepth);
GetDepth->UnUse( );
glBindFramebuffer(GL_FRAMEBUFFER, 0);
```

**In shadows.cpp:
Display( ), I**

In shadows.cpp:
Display( ), II

```cpp
RenderWithShadows->Use();
RenderWithShadows->SetUniformVariable((char*)"uShadowMap", 0 );
RenderWithShadows->SetUniformVariable((char*)"uLightX", LightX);
RenderWithShadows->SetUniformVariable((char*)"uLightY", LightY);
RenderWithShadows->SetUniformVariable((char*)"uLightZ", LightZ);
RenderWithShadows->SetUniformVariable((char*)"uLightSpaceMatrix", lightSpaceMatrix);

glm::vec3 eye = glm::vec3(0., 0., 8.);
glm::vec3 look = glm::vec3(0., 0., 0.);
glm::vec3 up = glm::vec3(0., 1., 0.);
glm::mat4 modelview = glm::lookAt(eye, look, up);

if (Scale < MINSCALE)          Scale = MINSCALE;
glm::vec3 scale = glm::vec3(Scale, Scale, Scale);
modelview = glm::scale(modelview, scale);
glm::vec3 xaxis = glm::vec3(1., 0., 0.);
glm::vec3 yaxis = glm::vec3(0., 1., 0.);
modelview = glm::rotate(modelview, glm::radians(Yrot), yaxis);
modelview = glm::rotate(modelview, glm::radians(Xrot), xaxis);
RenderWithShadows->SetUniformVariable((char*)"uModelView", modelview);

glm::mat4 proj = glm::perspective(glm::radians(75.f), 1.f, .1f, 100.f);
RenderWithShadows->SetUniformVariable((char*)"uProj", proj);
DisplayOneScene(RenderWithShadows);
RenderWithShadows->UnUse( );
```

Computer Graphics

OpenGL Demo Program: Rendering using the Depth Shadow Map

GetDepth.vert

```
#version 330 compatibility

uniform mat4 uLightSpaceMatrix;
uniform mat4 uAnim;

void
main()
{
    gl_Position = uLightSpaceMatrix * uAnim * gl_Vertex;
}
```

GetDepth.frag

```
#version 330 compatibility

uniform vec3 uColor;

void main()
{
    gl_FragColor = vec4(uColor, 1.);    // really doesn't matter...
}
```

Oregon State
University
Computer Graphics

```glsl
#version 330 compatibility

uniform mat4   uLightSpaceMatrix;
uniform mat4   uAnim;
uniform mat4   uModelView;
uniform mat4   uProj;
uniform float   uLightX;
uniform float   uLightY;
uniform float   uLightZ;


out vec4 vFragPosLightSpace;
out vec3 vNs;
out vec3 vLs;
out vec3 vEs;


void
main()
{
        vec3 LightPosition = vec3(uLightX, uLightY, uLightZ);

        vec4 ECposition = uModelView * uAnim * gl_Vertex;
        vec3 tnorm = normalize( mat3(uAnim) * gl_Normal );
        vNs = tnorm;
        vLs = LightPosition      - ECposition.xyz;
        vEs = vec3( 0., 0., 0. ) - ECposition.xyz;

        vFragPosLightSpace = uLightSpaceMatrix      *   uAnim * gl_Vertex;
        gl_Position               = uProj * uModelView      * uAnim * gl_Vertex;
}
```

RenderWithShadows.vert

```
#version 330 compatibility

uniform vec3          uColor;
uniform sampler2D uShadowMap;
uniform int            uShadowsOn;

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 = fragPosLightSpace.xyz / fragPosLightSpace.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;
}
```

RenderWithShadows.frag, I

Co

RenderWithShadows.frag, II

```
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 cnsidered 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( lighting, 1. );
}
```
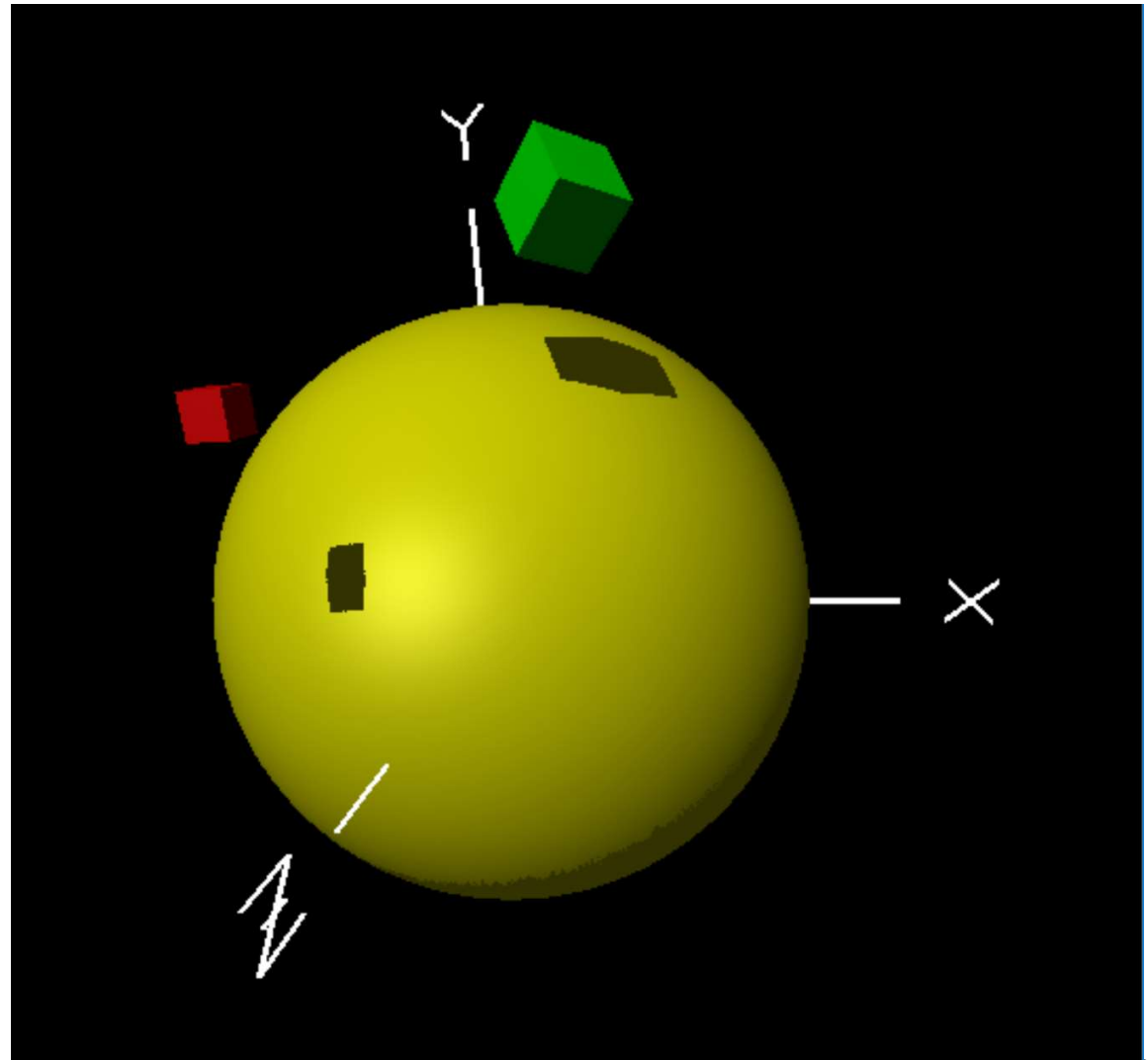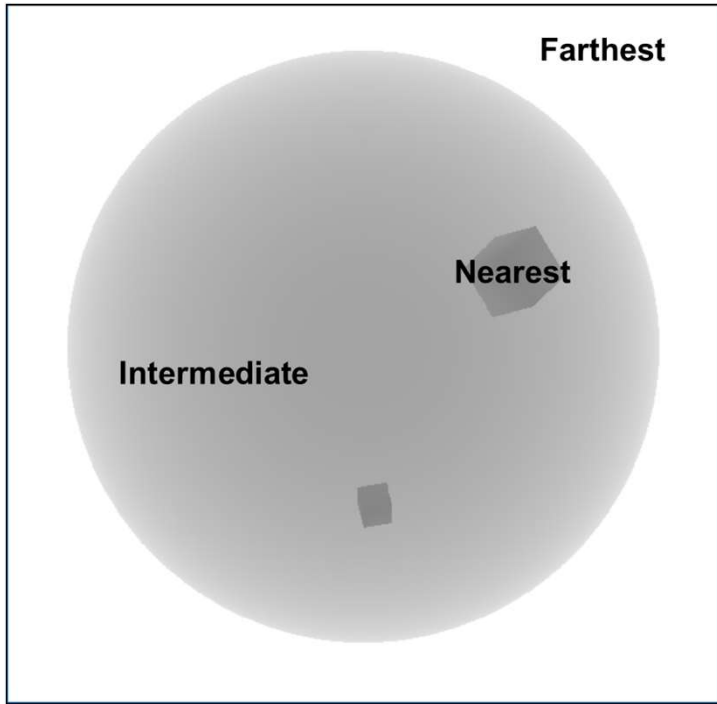
Computer Graphics

OpenGL Demo Program: Rendering into the Depth Shadow Map

In shadows.cpp

```
DisplayShadowMap->Use();
DisplayShadowMap->SetUniformVariable((char*)"uShadowMap", 0 );

glm::mat4 model = glm::mat4(1.f);
DisplayShadowMap->SetUniformVariable((char*)"uModel", model);

glm::vec3 eye = glm::vec3(0., 0., 1.);
glm::vec3 look = glm::vec3(0., 0., 0.);
glm::vec3 up = glm::vec3(0., 1., 0.);
glm::mat4 view = glm::lookAt(eye, look, up);
DisplayShadowMap->SetUniformVariable((char*)"uView", view);

glm::mat4 proj = glm::ortho(-0.6f, 0.6f,  -0.6f, 0.6f, .1f, 100.f);
DisplayShadowMap->SetUniformVariable((char*)"uProj", proj);

glBegin(GL_QUADS);
glTexCoord2f(0., 0.);
glVertex3f(-1., -1., 0.);
glTexCoord2f(1., 0.);
glVertex3f( 1., -1., 0.);
glTexCoord2f(1., 1.);
glVertex3f( 1.,  1., 0.);
glTexCoord2f(0., 1.);
glVertex3f(-1., 1., 0.);
glEnd();

DisplayShadowMap->UnUse( );
```
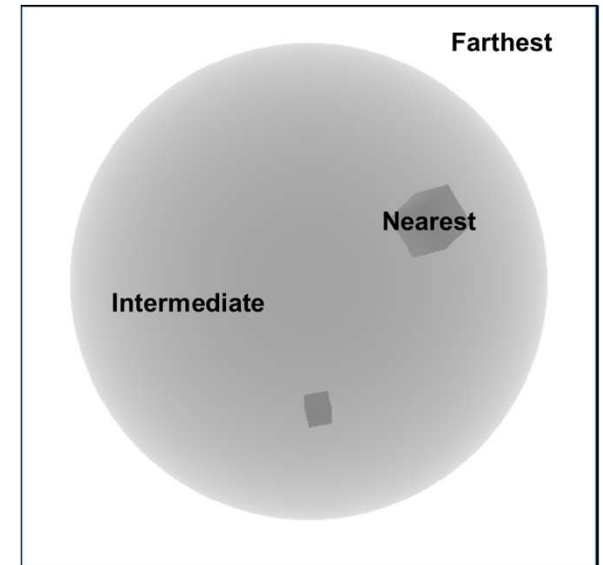

University
Computer Graphics

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

```
#version 330 compatibility

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.vert


```
#version 330 compatibility

uniform sampler2D uShadowMap;

in vec2 vST;

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

DisplayShadowMap.frag