Shadow Maps

- Agenda:
  - Assignments:
    - Discuss assignment #1...some more
    - Discuss assignment #2...anyone started?
  - First quiz!
  - Presentation of reading
  - Introduce shadow maps
  - Work on second programming assignment

Shadow texture recap

- Draw to a texture with the light as the eye.
  - Areas where a caster is visible are in shadow on the receiver
- Draw the texture on the receiver
  - Use projective texturing to apply shadow to non-planar objects
  - Battlefield 1942 does this (see image at right)

Shadow maps

- Shadow maps are remarkably similar
- Draw *depth values* to a texture with the light as the eye
- Draw the texture on the receiver
  - Use projective texturing
  - Use texgen (or shader) to store distance to light in the *R* component
  - Use special texture look-up function
    - If *R* is greater than texture value, fragment is in shadow

Depth textures

- Textures can store depth values instead of color values.
  - Added in OpenGL 1.4 or via several extensions.
- Depth textures have a base internal format of `GL_DEPTH_COMPONENT`.
  - Different internal formats specify number of depth bits (e.g., `GL_DEPTH_COMPONENT24` has 24 bits).
- Depth textures data specified via `glTexImage`, `glCopyTexImage`, etc.

Depth textures with FBOs

- Depth textures can be attached to an FBO just like a color texture:

```
glBindTexture(GL_TEXUTRE_2D, depth_tex);
glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT24,
    h, w, 0, GL_DEPTH_COMPONENT, GL_UNSIGNED_INT,
    NULL);
/* set other texture parameters. */

glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, fbo);
glFramebufferTexture2DEXT(GL_FRAMEBUFFER_EXT,
    GL_DEPTH_ATTACHMENT_EXT, GL_TEXTURE_2D,
    depth_tex, 0);
/* set color attachments. */
```

Projective texturing

- Does what it says: projects a texture onto an object
  - Used this first term to for a complex shaped spot light.
- In fixed-function:
  - Use `GL_EYE_LINEAR` texgen to calculate *S*, *T*, and *Q* texture coordinates
  - Use perspective projection texture matrix with the projector point as the “eye”
  - Final texture coordinates are *S/Q* and *T/Q*

Projective texturing in GLSL

- The GL spec says what the texgen math is, so we can implement it in GLSL.
- Ditto for texture matrix.
- In fact, we can access the fixed-function state using built-in uniforms:
  - `gl_TextureMatrix[gl_MaxTextureCoords]`
  - `gl_EyePlaneS[gl_MaxTextureCoords]`, etc.
  - Makes it easy to convert fixed-function code to GLSL.

Projective texturing in GLSL (cont.)

- Projective texturing uses different sampler functions in GLSL:
  - `texture1DProj` vs. `texture1D`
  - `texture2DProj` vs. `texture2D`
  - `texture3DProj` vs. `texture3D`
  - No cubic textures. Why?
  - Use these functions instead of doing the divide by hand!

Shadow map sampling

- In GLSL, shadow maps have different samplers...
  - `sampler1DShadow`
  - `sampler2DShadow`
  - No 3D or cubic textures! Why?
- And different texture functions...
  - `shadow1D`, `shadow1DProj`
  - `shadow2D`, `shadow2DProj`
  - Use these instead of doing the compare by hand!

Questions?

Legal Statement

- This work represents the view of the authors and does not necessarily represent the view of IBM or the Art Institute of Portland.
- OpenGL is a trademark of Silicon Graphics, Inc. in the United States, other countries, or both.
- Khronos and OpenGL ES are trademarks of the Khronos Group.
- Other company, product, and service names may be trademarks or service marks of others.
- Image from Battlefield 1942 is © Copyright Digital Illusions CE 2002.