Planar Shadows

- Agenda:
  - Discuss assignment #1
  - Wrap up render-to-texture techniques
    - Framebuffer objects
  - Introduce shadows
    - Importance of shadows
    - Planar shadows
    - Soft shadows
  - Start second programming assignment

FBOs

- Six steps to using a framebuffer object for render-to-texture:

FBOs

- Six steps to using a framebuffer object for render-to-texture:
  1. Create and bind the FBO.

FBOs

- Six steps to using a framebuffer object for render-to-texture:
  1. Create and bind the FBO.
  2. Attach textures and renderbuffers.
     - Must attach all needed buffers (e.g., color buffer and depth buffer).

FBOs

- Six steps to using a framebuffer object for render-to-texture:
  1. Create and bind the FBO.
  2. Attach textures and renderbuffers.
  3. Validate FBO.

FBOs

- Six steps to using a framebuffer object for render-to-texture:
  1. Create and bind the FBO.
  2. Attach textures and renderbuffers.
  3. Validate FBO.
  4. Render to FBO.

FBOs

- Six steps to using a framebuffer object for render-to-texture:
  1. Create and bind the FBO.
  2. Attach textures and renderbuffers.
  3. Validate FBO.
  4. Render to FBO.
  5. Unbind FBO.

FBOs

- Six steps to using a framebuffer object for render-to-texture:
  1. Create and bind the FBO.
  2. Attach textures and renderbuffers.
  3. Validate FBO.
  4. Render to FBO.
  5. Unbind FBO.
  6. Use textures.

FBO Creation

- An FBO object ID is created much like a texture:

```
glGenFramebuffersEXT(1, &fbo);
```

  - You can also assign your own object ID, just like with a texture.

- After the object ID is assigned, the FBO is bound for editing or use like a texture:

```
glBindFramebufferEXT(GL_FRAMEBUFFER_EXT,
  fbo);
```

Attaching Textures

- Textures are attached to an FBO using a function that matches the dimensionality of the texture:
  - `glFramebufferTexture1DEXT` – Attach a 1D texture.
  - `glFramebufferTexture2DEXT` – Attach a 2D texture or a cube map face.
  - `glFramebufferTexture3DEXT` – Attach a slice of a 3D texture.

Attaching Renderbuffers

- Created using `glGenRenderbuffersEXT` and `glRenderbufferStorageEXT`.
  - Analogous to `glGenTextures` and `glTexImage2D`.
  - Renderbuffers are renderable, but *not* texturable.
  - Only way to supply data to a renderbuffer is by rendering to it.
- Attach to FBO using `glFramebufferRenderbufferEXT`.

Validating an FBO

- Once all of the attachments have been made, the FBO must be validated.
  - `glCheckFramebufferStatusEXT`
- There are 3 classes of return values from `glCheckFramebufferStatusEXT`:

Validating an FBO

- Once all of the attachments have been made, the FBO must be validated.
  - `glCheckFramebufferStatusEXT`
- There are 3 classes of return values from `glCheckFramebufferStatusEXT`:
  - Success – the FBO can be used for rendering.

Validating an FBO

- Once all of the attachments have been made, the FBO must be validated.
  - `glCheckFramebufferStatusEXT`
- There are 3 classes of return values from `glCheckFramebufferStatusEXT`:
  - Success
  - Implementation independent error – these are always errors and represent a bug in the application.

Validating an FBO

- Once all of the attachments have been made, the FBO must be validated.
  - `glCheckFramebufferStatusEXT`
- There are 3 classes of return values from `glCheckFramebufferStatusEXT`:
  - Success
  - Implementation independent error
  - Implementation dependent error – the hardware can't handle the combination of attachments, etc.

Validating an FBO

- Common device independent errors:

Validating an FBO

- Common device independent errors:
  - Attached texture is incomplete
    - `GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT_EXT`

Validating an FBO

- Common device independent errors:
  - Attached texture is incomplete
  - Dimensions of attachments do not match
    - `GL_FRAMEBUFFER_INCOMPLETE_DIMENSIONS_EXT`

Validating an FBO

- Common device independent errors:
  - Attached texture is incomplete
  - Dimensions of attachments do not match
  - Nothing is attached to the FBO
    - `GL_FRAMEBUFFER_INCOMPLETE_MISSING_ATTACHMENT_EXT`

Validating an FBO

- Common device independent errors:
  - Attached texture is incomplete
  - Dimensions of attachments do not match
  - Nothing is attached to the FBO
  - Attached color attachments have mismatched formats
    - `GL_FRAMEBUFFER_INCOMPLETE_FORMATS_EXT`

Validating an FBO

- Common device independent errors:
  - Attached texture is incomplete
  - Dimensions of attachments do not match
  - Nothing is attached to the FBO
  - Attached color attachments have mismatched formats
  - Missing color attachment for named draw buffer
    - `GL_FRAMEBUFFER_INCOMPLETE_DRAW_BUFFER_EXT`

Validating an FBO

- Common device independent errors:
  - Attached texture is incomplete
  - Dimensions of attachments do not match
  - Nothing is attached to the FBO
  - Attached color attachments have mismatched formats
  - Missing color attachment for named draw buffer
  - Missing color attachment and read buffer is not `NONE`
    - `GL_FRAMEBUFFER_INCOMPLETE_READ_BUFFER_EXT`

Example

```
glBindTexture(GL_TEXTURE_2D, 2);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA8, 256, 256, 0,
    GL_RGBA, GL_UNSIGNED_BYTE, NULL);

glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, 1);
glFramebufferTexture2DEXT(GL_FRAMEBUFFER_EXT,
    GL_COLOR_ATTACHMENT0_EXT, GL_TEXTURE_2D, 2, 0);

GLenum fbo_status =
     glCheckFramebufferStatusEXT(GL_FRAMEBUFFER_EXT);
if (fbo_status != GL_FRAMEBUFFER_COMPLETE_EXT) {
    /* error */
}
```

Render to FBO

- FBO rendering is enabled whenever a non-zero FBO is bound.
  - Just like program objects.
- May need to reset the viewport .
- Draw just like normal.
- When done rendering to the FBO, bind the 0 object.

Example

```
glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, my_fbo);

glGetFramebufferAttachmentParameterivEXT(
    GL_FRAMEBUFFER_EXT, GL_COLOR_ATTACHMENT0_EXT,
    GL_WIDTH, &width);
glGetFramebufferAttachmentParameterivEXT(
    GL_FRAMEBUFFER_EXT, GL_COLOR_ATTACHMENT0_EXT,
    GL_HEIGHT, &height);

glViewport(0, 0, width, height);

/* Draw */

glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, 0);
```

Use the texture

- After unbinding the FBO, use the texture just like normal.
  - No, really!
- Don't try to render to a mipmap level that is selected for rendering.
  - Results are undefined, but probably not what you would want anyway.
  - More on this in a moment...

Mipmaps

- Two ways to generate mipmaps with FBO render-to-texture
  - Use the explicit mipmap generation routine, `glGenerateMipmapEXT`, after rendering and unbinding FBO.
  - Generate the mipmaps by rendering to the other mipmap levels!
    - Have to clamp the texture LOD.
    - Especially useful for mipmapping normal maps…remember the paper from last term?

Example

```
glBindTexture(GL_TEXTURE_2D, tex);
glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, fbo);

for (unsigned lod = 1; lod < levels; lod++) {
    glTexParameteri(GL_TEXTURE_2D,
        GL_TEXTURE_BASE_LEVEL, lod - 1);
    glTexParameteri(GL_TEXTURE_2D,
        GL_TEXTURE_MAX_LEVEL, lod - 1);

    glFramebufferTexture2DEXT(GL_FRAMEBUFFER_EXT,
        GL_COLOR_ATTACHMENT0_EXT, GL_TEXTURE_2D,
        tex, lod);

    /* Draw */
}
```

Shadows

- Why are shadows important in 3D rendering?

Shadows

- Why are shadows important in 3D rendering?
  - Give cues about shadow casters
    - Relative positions of casters
    - Relative positions of caster and receiver

Shadows

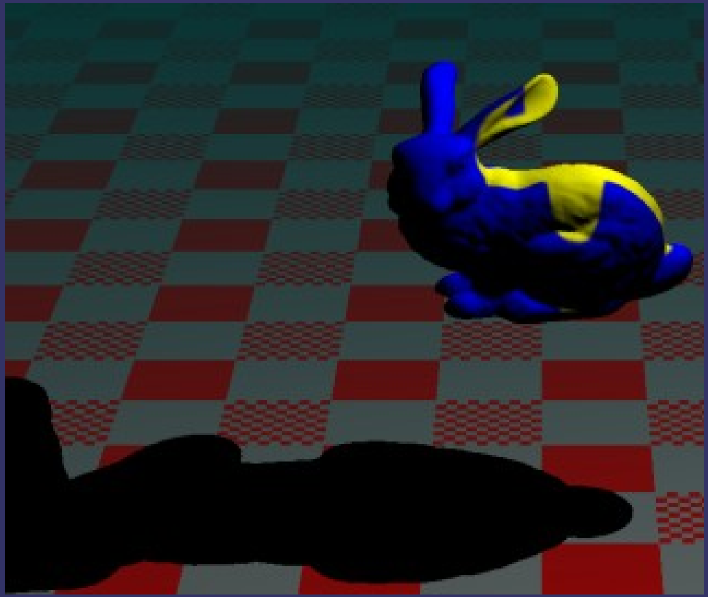
- Why are shadows important in 3D rendering?
  - Give cues about shadow casters
    - Relative positions of casters
    - Relative positions of caster and receiver
  - Give cues about shadow receivers
    - Show additional surface detail

Shadow terms

- Receiver – object that is shadowed
- Caster – object that blocks light from the receiver
  - May also be called *occluder* because it occludes the light from the receiver
- Umbra – Region on receiver that is completely shadowed
- Penumbra – Transition region between umbra and non-shadowed area

Planar Shadows

- Simplest shadows are those projected onto a flat plane
  - As the description implies, this can be done using a projection matrix



Plane equation

- Give a point on a plane, *p*, and the normal of that plane, *n*, calculate the plane equation:

$$d = -(n \cdot p)$$

$$n \cdot p_i + d = 0$$

Projection onto a plane

- Given a plane, defined by *n* and *d,* and a projection point, *p*, create a matrix that projects an arbitrary point onto that plane.
  - Like the projection of the view plane and the eye point.

$$M=\begin{bmatrix} n\cdot p+d-p_x n_x & -p_x n_y & -p_x n_z & -p_x d \\ -p_y n_x & n\cdot p+d-p_y n_y & -p_y n_z & -p_y d \\ -p_z n_x & -p_z n_y & n\cdot p+d-p_z n_z & -p_z d \\ -n_x & -n_y & -n_z & n\cdot p \end{bmatrix}$$

Planar shadows

- If the plane is the ground plane, and the projection point is the light, M is a matrix that projects the shadow of world-space geometry onto the ground.
- But where do we insert M into the transformation stack?

Planar shadows

- If the plane is the ground plane, and the projection point is the light, *M* is a matrix that projects the shadow of world-space geometry onto the ground.
- But where do we insert *M* into the transformation stack?
  - After the object-to-world space transformations, but before the world-to-eye space transformation.

Drawing a planar shadow

- Many possible methods. Here's one that works:
  - Disable depth buffer
    - `glDepthMask(GL_FALSE);`
  - Draw shadow to alpha buffer
    - `glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_TRUE);`
  - Enable depth buffer
  - Draw object
  - Draw ground and modulate with destination alpha
    - ```
      glEnable(GL_BLEND);
      glBlendFunc(GL_ONE_MINUS_DST_ALPHA, GL_ONE);
      ```
      ```

Hard shadows are better than nothing, but not much!

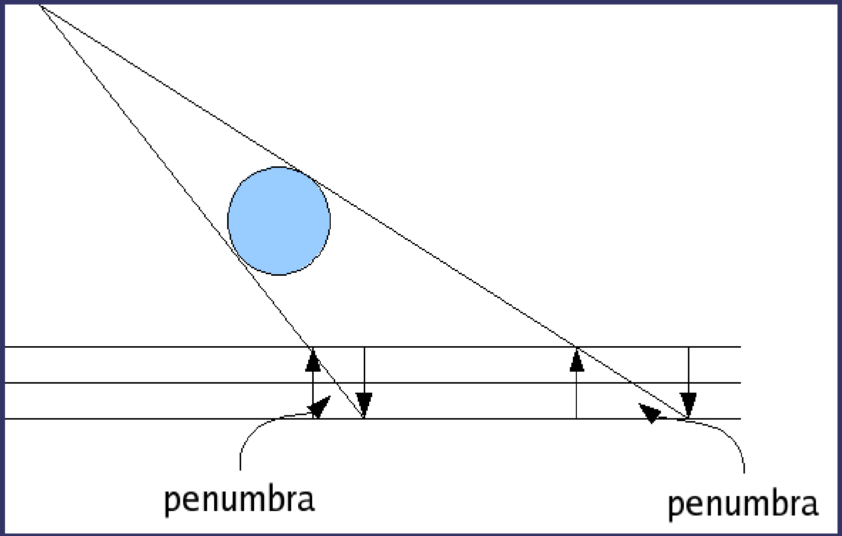
- Can this technique be extended to create soft shadows?
  - Soft shadows are created when the light has “area”
  - An LED in a dark room casts only hard shadows

Heckbert and Herf's method

- Simulate an area light with many point lights on the area light's surface
  - If *lots* of sample points are used, this method produces *very* good results
  - If *lots* of sample points are used, this method produces *very* slow results
  - Some optimizations are possible
    - Scale number of samples with size of light
    - Scale number of samples with distance between light and shadow caster

Gooch's method

- By moving the receiving plane towards and away from the light, the penumbra can be simulated
  - The simulated penumbra is *always* too big
- After projecting onto an offset plane, the projectiong has to be moved to the correct plane.



Shadow textures

- One way to implement Gooch's method is to render the shadow to a texture, then draw the shadow texture multiple times.
  - Draw this texture with the light as the eye.
- Can just use a single pass and linear filtering
  - Use projective texturing to apply shadow to non-planar objects
  - Battlefield 1942 does this (see image at right)

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.