VGP353 – Week 3

- Agenda:
  - Quiz #1
  - Assignment #1 due
  - Introduce shadow maps
    - Differences / similarities with shadow textures
    - Added benefits
    - Potential problems

15-April-2008

Shadow Textures

- As discussed last week, shadow textures have a number of faults
  - Separate texture for each caster / light pair
  - No self-shadowing
  - Difficulty with casters / receivers that a nearly the same distance from the light
- What is the fundamental limitation at the root of all these problems?

Shadow Textures

- As discussed last week, shadow textures have a number of faults
  - Separate texture for each caster / light pair
  - No self-shadowing
  - Difficulty with casters / receivers that a nearly the same distance from the light
- What is the fundamental limitation at the root of all these problems?
  - Each shadow texel is a simple on-or-off. The remaining information must be inferred.

Shadow Textures

- To determine whether a position in 3-space is in shadow, what information is needed?

Shadow Textures

- To determine whether a position in 3-space is in shadow, what information is needed?
  - Is there something *closer* to the light in the direct line-of-sight
    - The shadow texture only tells whether there is something in the line of sight, *not* whether that something is closer to the light

Shadow Maps

- Instead of storing boolean “shadow” / “not shadow”, store the distance from the light to the closest shadow caster
  - This is a *shadow map*
  - Compare the distance read from the shadow map to the distance between the object and the light
    - If $distance_{shadow} < distance_{object}$, the fragment is in shadow
    - If $distance_{shadow} \geq distance_{object}$, the fragment is not in shadow

Shadow Maps

- Shadow map stores “distance to nearest shadow caster.”
  - *Remind* you of anything?

Shadow Maps

- Shadow map stores “distance to nearest shadow caster.”
  - *Remind* you of anything?
  - A depth buffer!
    - Depth buffer (typically) stores the per-pixel distance to the object nearest to the eye
    - When rendering from the light's PoV, the distance stored in the depth buffer is the distance to the object nearest to the light

Shadow Textures vs. Shadow Maps

- Shadow texture:
  - Draw either light color or shadow color to a color texture
  - Read light color directly from shadow texture
  - Color fragment based on light color
- Shadow map:
  - Draw distance to nearest object to a depth texture
  - Compare occluder distance to object distance
  - Color fragment base on result of comparison

Shadow Maps

- Advantages:

Shadow Maps

- Advantages:
  - Objects can self-shadow!
  - Near-by objects can shadow each other correctly

Shadow Maps

- Advantages:
  - Objects can self-shadow!
  - Near-by objects can shadow each other correctly
- Disadvantages:

Shadow Maps

- Advantages:
  - Objects can self-shadow!
  - Near-by objects can shadow each other correctly
- Disadvantages:
  - Separate texture for each caster / light pair

Shadow Maps

- Advantages:
  - Objects can self-shadow!
  - Near-by objects can shadow each other correctly
- Disadvantages:
  - Separate texture for each caster / light pair
    - *Is this necessary? NO!*

Shadow Maps Revised

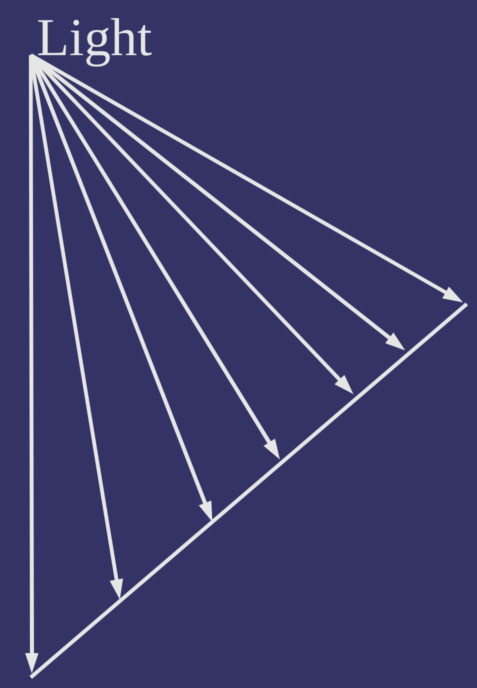
- Algorithm:
  - Group potential casters and receivers
  - Calculate frustum that encompasses all objects within a group
  - Render objects using calculate frustum. Store depth buffer in a texture (shadow map)
  - Render objects from the camera's PoV with appropriate shadow map. Use comparison previously described.

Shadow Map Problems

- Three big problems with shadow maps:
  - Sampling differences between shadow map rendering and reading...the dreaded “shadow acne”
  - Aliasing
  - Lack of depth precision
  - Omni-directional lights inside the view frustum

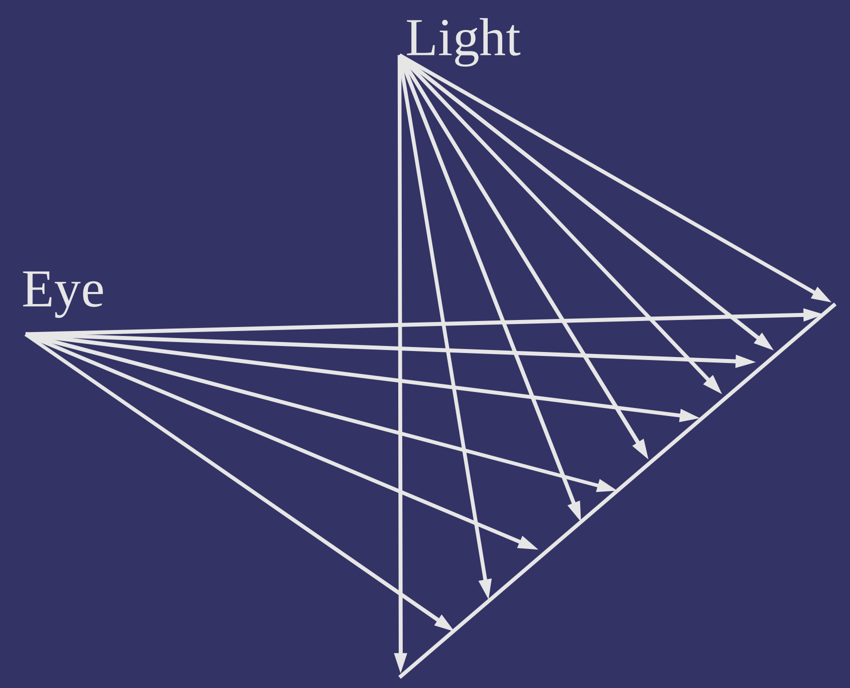
Shadow Acne

- Light and camera sample object at different positions
  - Drawing from the light's PoV samples one set of positions



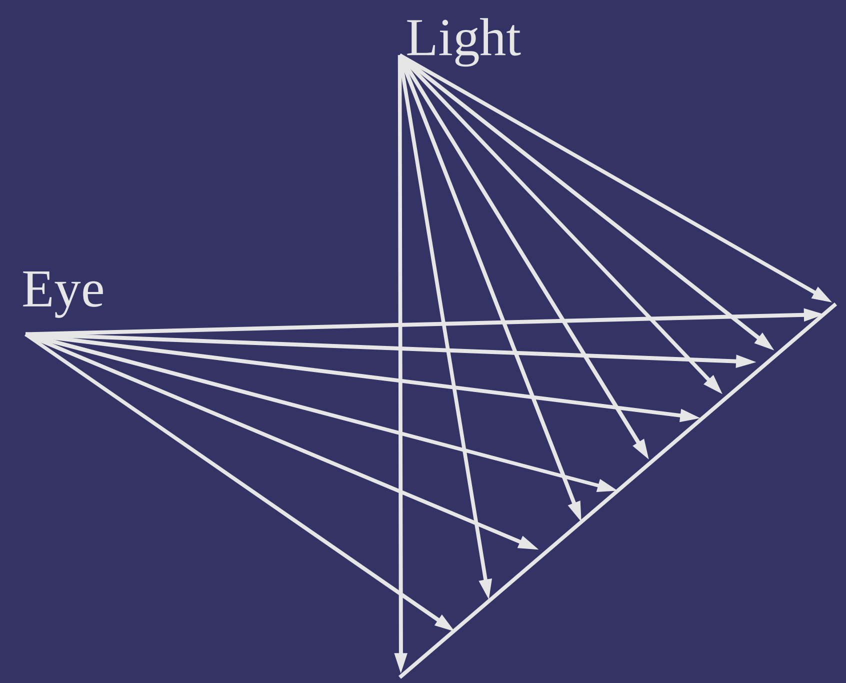
Shadow Acne

- Light and camera sample object at different positions
  - Drawing from the light's PoV samples one set of positions
  - Drawing from the camera's PoV samples a different set of positions

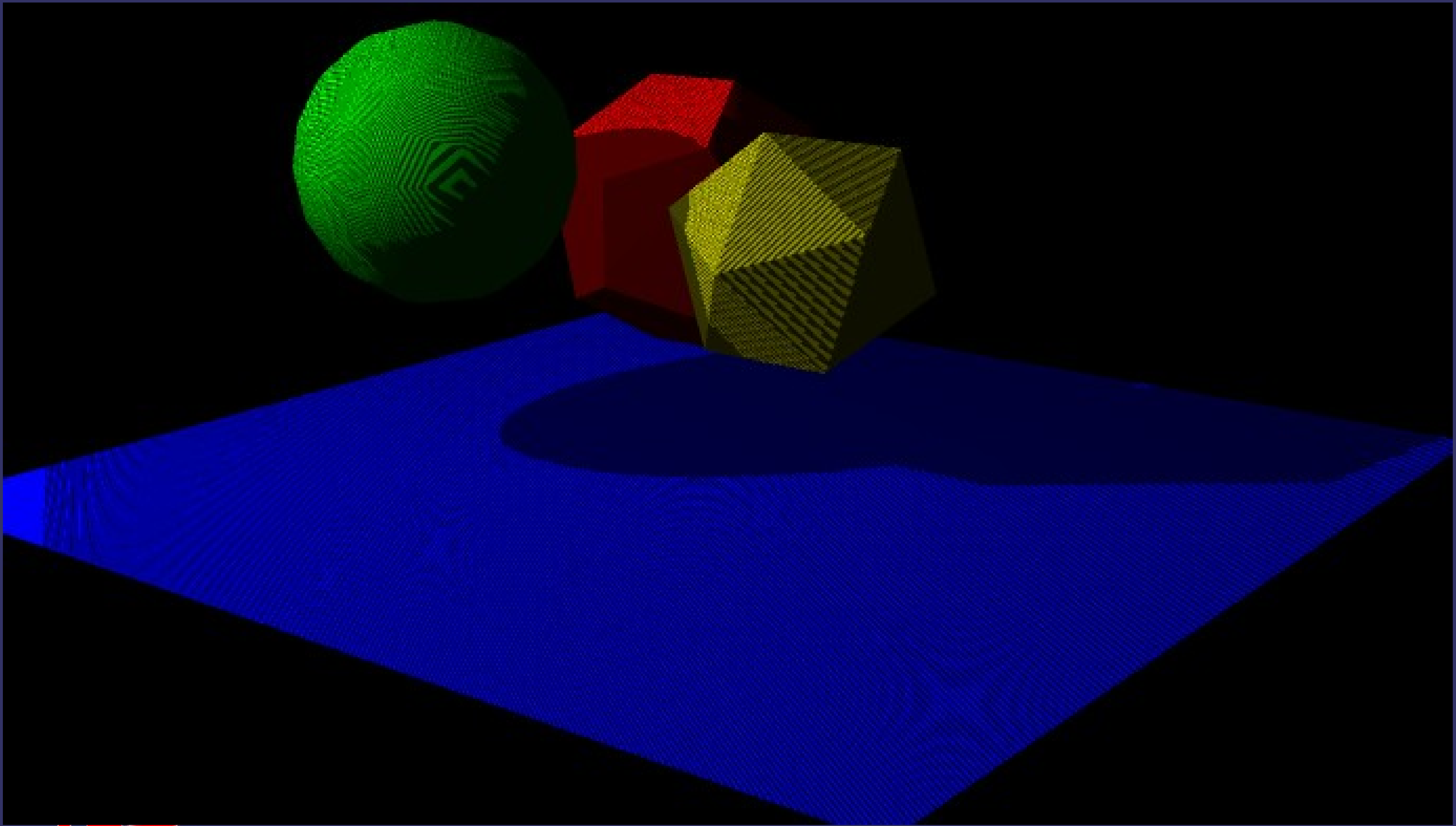


Shadow Acne

- Light and camera sample object at different positions
  - Drawing from the light's PoV samples one set of positions
  - Drawing from the camera's PoV samples a different set of positions
  - Result: incorrect values are used to determine if a surface shadows itself



Shadow Acne



15-April-2008

Shadow Acne

- Two common solutions:

Shadow Acne

- Two common solutions:
  - Render back faces to shadow map
    - Front faces aren't drawn to shadow map, so they won't self-shadow
    - Back faces aren't lit: depth comparison result is irrelevant

Shadow Acne

- Two common solutions:
  - Render back faces to shadow map
    - Front faces aren't drawn to shadow map, so they won't self-shadow
    - Back faces aren't lit: depth comparison result is irrelevant
  - Use polygon offset
    - Bias fragment depth by small factor to ensure $distance_{shadow} \geq distance_{object}$
      - `glPolygonOffset(1.1f, 1.0f);`
    - Very tricky to get right! Movie fx companies spend *lots* of time tweaking every frame to eliminate artifacts[1]

[1] G. King, "Shadow Mapping Algorithms." NVIDIA. 2004. ftp://download.nvidia.com/developer/presentations/2004/GPU_Jackpot/Shadow_Mapping.pdf

Shadow Map Aliasing

- Several sources of aliasing in shadow maps
  - Must use nearest-neighbor sampling
    - Straightforward bi-linear or mipmap sampling would average depth values together for use in comparison
  - Depth maps are typically small, so fine details may get lost
    - Shadows from thin objects (telephone wires, chain link fence, etc.) may disappear
    - Small gaps between objects may fill-in
  - Objects distant from light may be too small in shadow map
    - If the object's shadow is near the camera, it will appear very blocky

Shadow Map Precision

- Every Z-buffer has potential precision problems
  - Objects distant from near-plane get fewer significant bits to store depth
    - May not be noticeable far from the near plane
  - Due to viewing differences, lack of Z precision far from *light's* near-plane may result in artifacts close to *camera's* near-plane

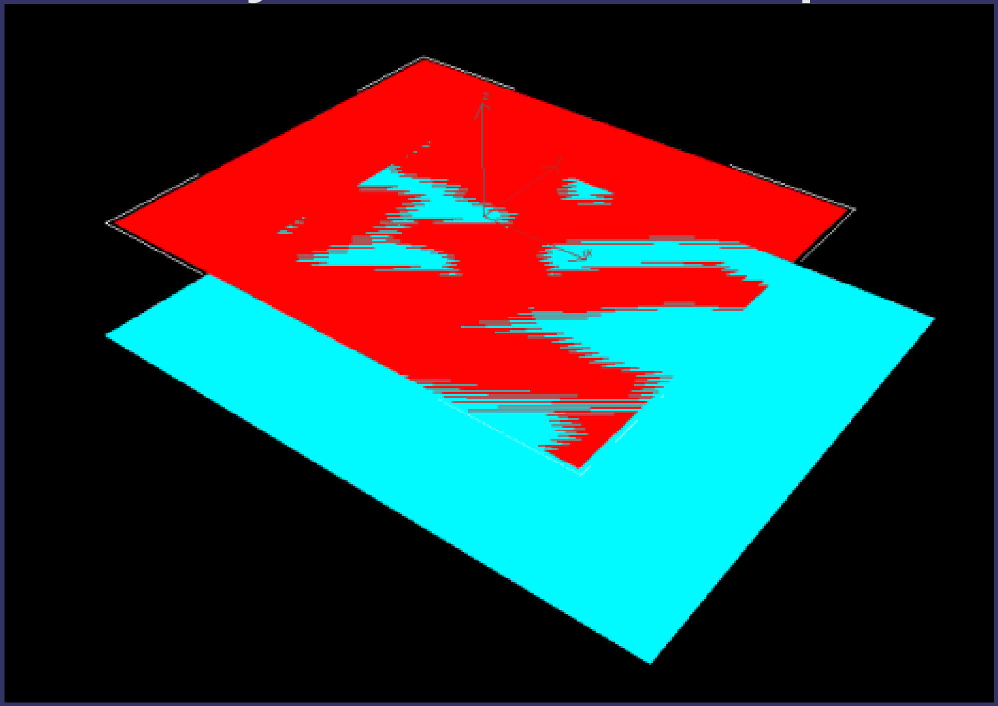
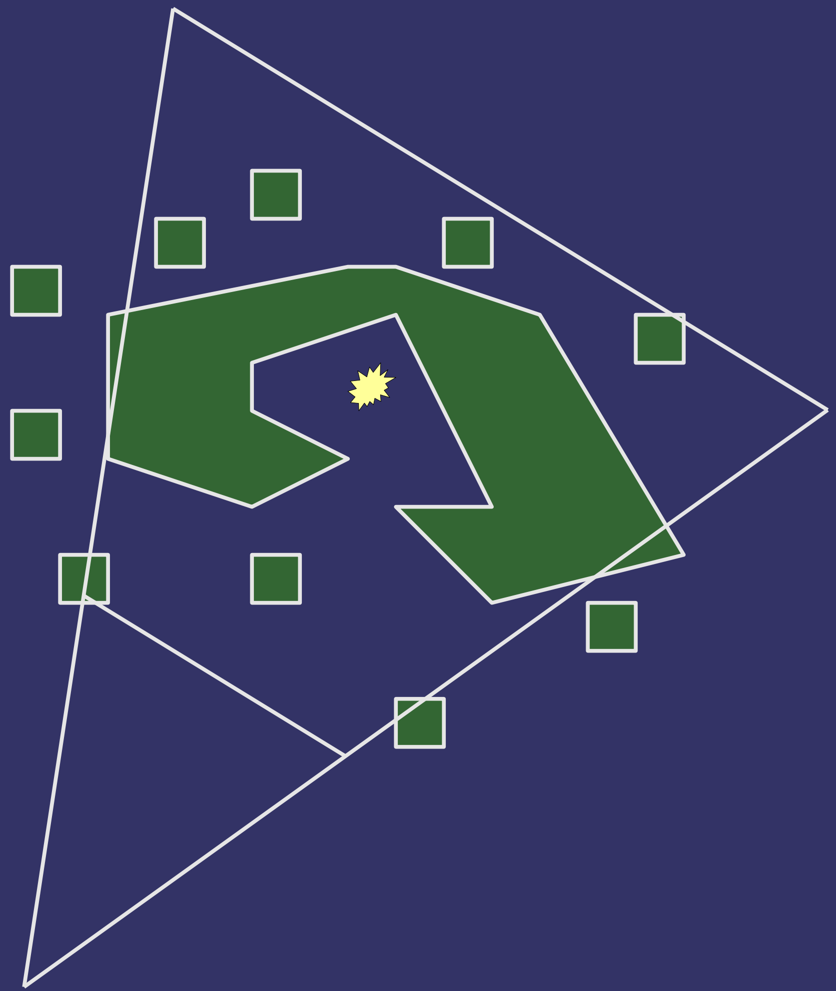


Image from http://en.wikipedia.org/wiki/Z-fighting

Omni-directional Lights

- Consider this scene...
  - What frustum do we pick for the light and the large object?
  - We'd need a 360° field-of-view!



Shadow Maps in GLSL

- New sampler types:
  - `sampler1DShadow` and `sampler2DShadow`
- New sampler functions:
  - `shadow1D` and `shadow1DProj`
  - `shadow2D` and `shadow2DProj`
    - 3rd component of texture coordinate is the distance used for comparison
  - As with projective textures, use shadow sampler types and functions instead of doing comparisons by hand

Shadow Maps in GLSL

- Each texture has a depth comparison mode
  - Mode is set by calling `glTexParameteri` with *name* of `GL_TEXTURE_COMPARE_FUNC`
  - Sets mode used for comparison in `sampler[12]D` functions
- Sampler function returns 1.0 if the test passes or 0.0 if the test fails

Depth Textures

- Store single component, normalized value used for depth (shadow) comparisons
  - Use one of three internal formats:
    - `GL_DEPTH_COMPONENT16`
    - `GL_DEPTH_COMPONENT24`
    - `GL_DEPTH_COMPONENT32`
  - Only format that can be used with GLSL shadow samplers
    - Can be also use with non-shadow samplers as a luminance, intensity, or alpha texture

Depth Textures

- Create just like any other texture:

```
glBindTexture(GL_TEXTURE_2D, my_shadow_tex);
glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT24, 0,
   0, width, height, GL_DEPTH_COMPONENT24,
   GL_UNSIGNED_INT, NULL);
```

Depth Textures

- Create just like any other texture:

```
glBindTexture(GL_TEXTURE_2D, my_shadow_tex);
glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT24, 0,
   0, width, height, GL_DEPTH_COMPONENT24,
   GL_UNSIGNED_INT, NULL);
```

- To use as false-color texture:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_MODE,
   GL_NONE);
glTexParameteri(GL_TEXTURE_2D, GL_DEPTH_TEXTURE_MODE,
   GL_INTENSITY);
```

  - Or GL_LUMINANCE or GL_ALPHA

Depth Textures

- Create just like any other texture:

```
glBindTexture(GL_TEXTURE_2D, my_shadow_tex);
glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT24, 0,
   0, width, height, GL_DEPTH_COMPONENT24,
   GL_UNSIGNED_INT, NULL);
```

- To use as false-color texture:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_MODE,
   GL_NONE);
glTexParameteri(GL_TEXTURE_2D, GL_DEPTH_TEXTURE_MODE,
   GL_INTENSITY);
```

  - Or GL_LUMINANCE or GL_ALPHA

- To use as a shadow map:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_MODE,
   GL_COMPARE_R_TO_TEXTURE);
```

Depth Textures and FBOs

- Attach the depth-component texture to the depth attachment:

```
glFramebufferTexture2DEXT(GL_FRAMEBUFFER_EXT,
   GL_DEPTH_ATTACHMENT_EXT, GL_TEXTURE_2D, tex, 0);
```

  - If there are no mipmaps (likely), as usual, be sure to set non-mipmap minification mode
  - If there is no color output (likely), be sure to disable all color buffer access:

```
glDrawBuffer(GL_NONE);
glReadBuffer(GL_NONE);
```

Next week...

- Advanced shadow map techniques
  - Percentage closer filtering
  - Depth range optimizations
  - Omni-directional lights

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.