VGP352 – Week 2

- Agenda:
  - Render to texture
  - Reflection mapping
    - Review
    - Rendering to a reflection map
  - Improving the reflection model
    - Reflection maps as better lights
    - Fresnel reflections

15-April-2009

Render to Texture

- Several methods exist
  - Render to framebuffer, the copy the result to a texture
    - Use `glCopyTexImage2D`
  - Render to a pixel buffer (pbuffer), then bind to a texture
    - Platform dependent (i.e., is different on Linux, Windows, and Mac OS)
  - Use framebuffer objects to render direct to a texture

Why render to a texture?

- Many effects can be created by rendering to one or more textures, then using those textures to render the final scene
  - Shadow maps
  - Dynamic environment maps
  - Pre-baking procedural textures

Copy to Texture

- Very easy:
  - Draw to backbuffer
  - Copy resulting image to a texture using either `glCopyTexImage2D` or `glCopyTexSubImage2D`
  - *That's it*

Copy to Texture

- Problems:
  - Must perform extra copies – slow
  - Must perform extra buffer clears
  - Window must be at least as large as the largest desired texture
  - Results can be corrupted if the window is partially obscured
  - Can't generate a texture when a frame is partially rendered
    - The back-buffer already has part of the final scene in it!

Framebuffer Objects

- Warning: FBOs have a fairly steep learning curve
  - The ARB spent over two years developing the interface
  - It builds on the familiar texture interfaces, but is still very different

Framebuffer Objects

- Create and bind an FBO

```
void glGenFramebuffersEXT(GLsizei n,
    GLuint *framebuffers);

void glBindFramebufferEXT(GLenum target,
    GLuint framebuffer);
```

Framebuffer Objects

- Attach one or more renderable objects to it
  - 1D, 2D, and 3D versions exist

```
void glFramebufferTexture2DEXT (GLenum target,
    GLenum attachment, GLenum textarget,
    GLuint texture, GLint level);

void glFramebufferRenderbufferEXT(
    GLenum target, GLenum attachment,
    GLenum renderbuffertarget,
    GLuint renderbuffer);
```

Framebuffer Objects

- Attach one or more renderable objects to it
  - 1D, 2D, and 3D versions exist

```
void glFramebufferTexture2DEXT (GLenum target,
    GLenum attachment, GLenum textarget,
    GLuint texture, GLint level);

void glFramebufferRenderbufferEXT(
    GLenum target, GLenum attachment,
    GLenum renderbuffertarget,
    GLuint renderbuffer);
```

Selects how the buffer is used:

- Color buffer: `GL_COLOR_ATTACHMENT0`
- Depth buffer: `GL_DEPTH_ATTACHMENT`
- Stencil buffer: `GL_STENCIL_ATTACHMENT`

Framebuffer Objects

- After making all of the desired attachments:
  - Disable outputs that don't have attachments
    - Use `glColorMask` or `glDisable` with `GL_DEPTH_TEST` or `GL_STENCIL_TEST`
  - Make sure the FBO is acceptable by calling

```
GLenum glCheckFramebufferStatusEXT(
    GLenum target);
```

    - Some hardware can't handle some combinations of attachments
    - Some combinations are just wrong
  - Reset the viewport
  - Draw!

Framebuffer Objects

- Use textures that were rendered to just like usual
  - You cannot render to a texture layer that might be used for rendering (i.e., no feedback loop)
  - You cannot use `GL_GENERATE_MIPMAPS` with FBO rendered textures

```
void glGenerateMipmapEXT(GLenum target);
```

Renderbuffers vs. Textures

- Two types of buffers can be attached to an FBO:
  - Texture – texturable and renderable
  - Renderbuffer – renderable only
- Why do renderbuffers exist?

Renderbuffers vs. Textures

- Two types of buffers can be attached to an FBO:
  - Texture – texturable and renderable
  - Renderbuffer – renderable only
- Why do renderbuffers exist?
  - It's the only way to do stencil... a “stencil texture” is a nonsensical concept
  - Driver may be able to use a better format if the object won't be texturable
    - Some hardware needs the whole mipmap stack allocated up-front

Renderbuffers

- Similar interface to textures:

```
void glGenRenderbuffersEXT(GLsizei n,
    GLuint *renderbuffers);

void glRenderbufferStorageEXT(GLenum target,
    GLenum internalformat,
    GLsizei width, GLsizei height);

void glDeleteRenderbuffersEXT(GLsizei n,
    const GLuint *renderbuffers);
```

Dimensions and Dimensionality

- Dimensions (i.e., height and width) of all attachments must match
  - This requirement is relaxed in OpenGL 3.0 and `GL_ARB_framebuffer_object`
- Dimensionality (i.e., 1D or 2D) of all attachments must match
  - A 2D “slice” of a 3D texture is attached, so it is treated as a 2D texture for this purpose

References

Jones, Rob, "OpenGL Framebuffer Object 101." http://www.gamedev.net/reference/programming/features/fbo1/

Green, Simon, The OpenGL Framebuffer Object Extension. NVIDIA. 2004. http://developer.nvidia.com/object/gdc_2005_presentations.html

`GL_EXT_framebuffer_object` and related extension specifications:

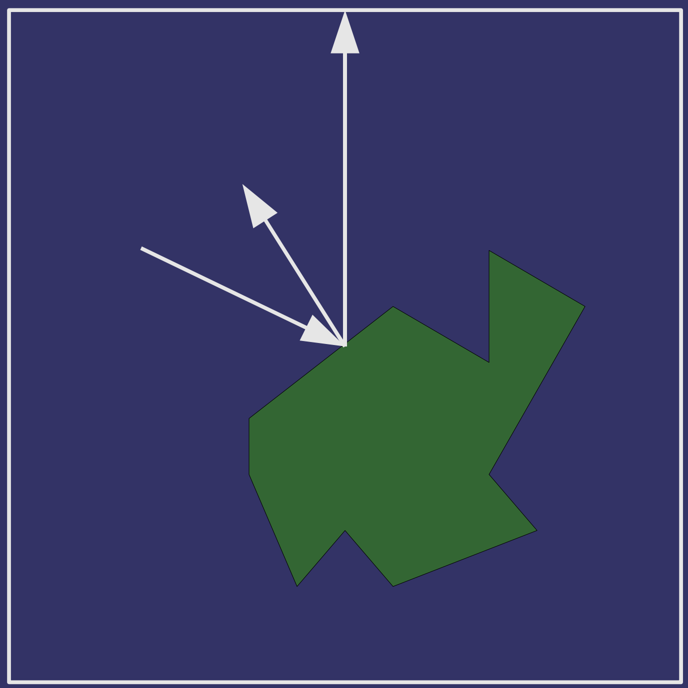
- http://www.opengl.org/registry/specs/EXT/framebuffer_object.txt
- http://www.opengl.org/registry/specs/EXT/framebuffer_blit.txt
- http://www.opengl.org/registry/specs/EXT/framebuffer_multisample.txt
- http://www.opengl.org/registry/specs/ARB/framebuffer_object.txt

Break

Reflection Mapping

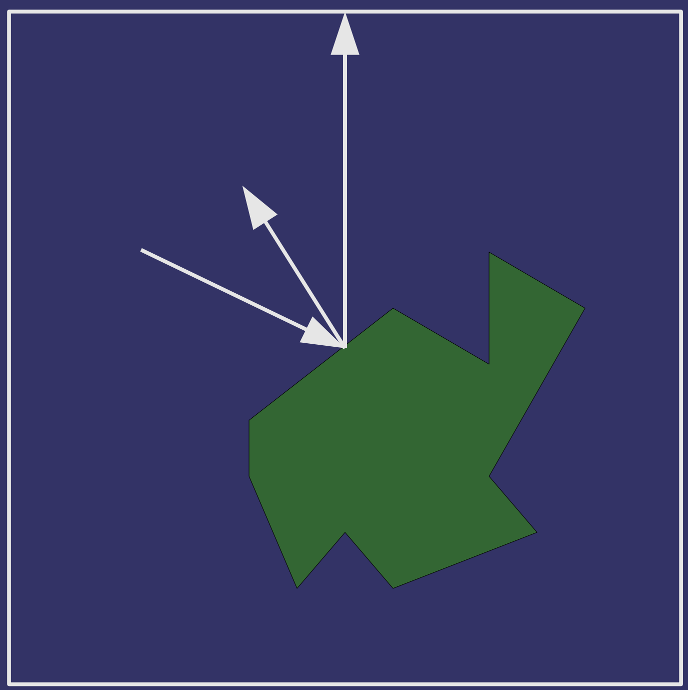
- Forms of reflection mapping are classified by the shape used to simulate the environment
  - Cylindrical
  - Hemispherical
  - Spherical
  - Cube
  - Dual-paraboloid

Reflection Mapping – Cube



- Extend R to intersect unit cube surrounding point

Reflection Mapping – Cube



- Pros:
  - Trivial to implement
  - Easy to render to reflection map
- Cons:
  - Requires hardware support
  - More difficult to get source images
  - Discontinuities at cube-face boundaries

Reflection Mapping – Cube

- From the point of view of the reflector:
  - Draw each of the 6 on-axis views to separate faces of the cube map
  - Be sure to pick a convenient "space" to draw in so that the reflection map can be used
    - Probably align the axes of the cube map to the world-space

Reflection Mapping – Paraboloid

- View of environment as reflected by a convex parabolic mirror
  - The *outside* of a satellite dish
  - Reflects 180° of the environment
    - Capture 360° by using two maps
    - Known as dual paraboloid
  - Fairly similar to a hemispherical reflection map

Reflection Mapping – Paraboloid

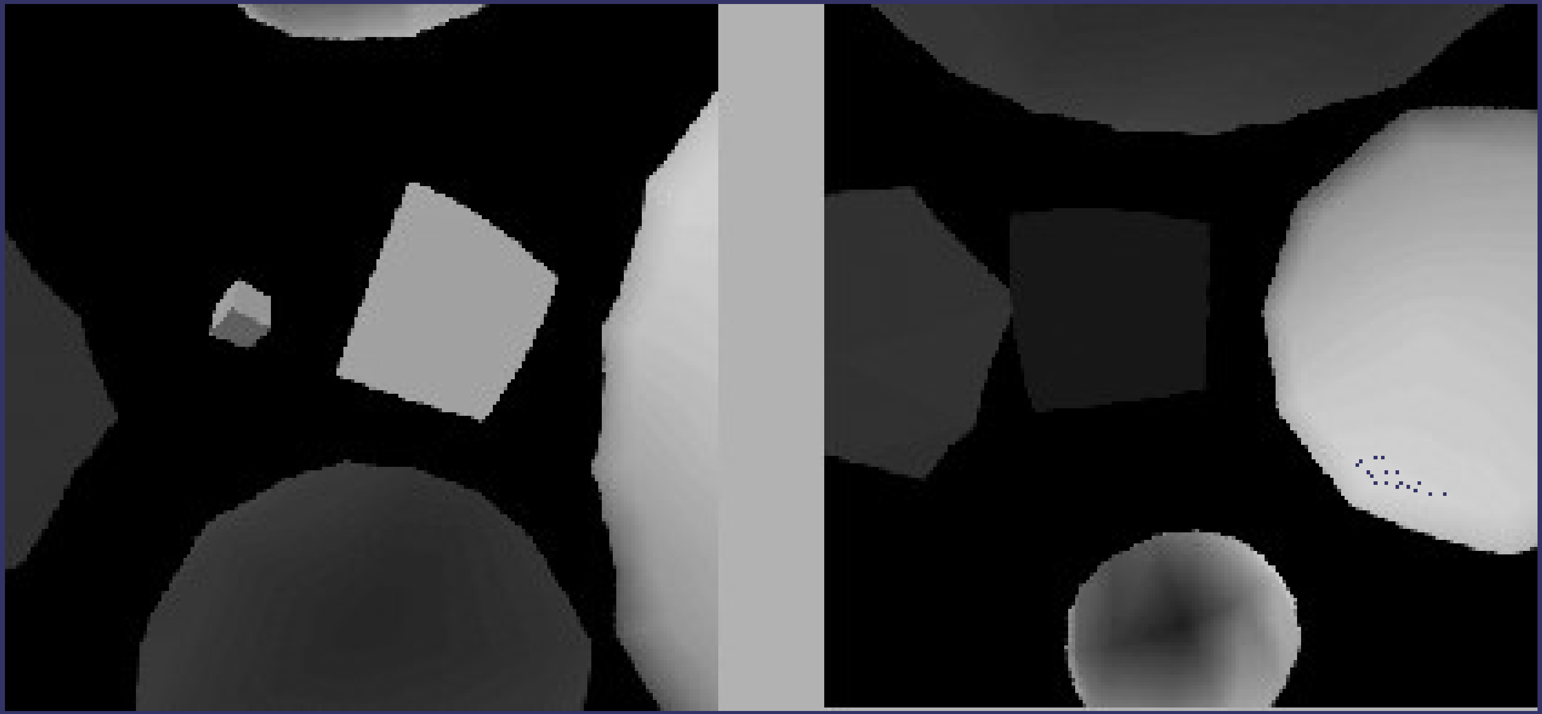
- Easily convert reflection vector to 2D texture coordinate for paraboloid map:

$$\begin{pmatrix} s \\ t \\ 1 \\ 1 \end{pmatrix} = A \cdot P \cdot S \cdot M_n^T \cdot R^T$$

$$A = \begin{pmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, S = \begin{pmatrix} -1 & 0 & 0 & d_x \\ 0 & -1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  - d is the view direction vector
    - { 0 0 1 } or { 0 0 -1 } depending on the viewing direction
  - M_n is the transformation matrix for normals

Reflection Mapping – Paraboloid



Original image from http://opengl.org/resources/code/samples/sig99/advanced99/notes/node185.html

Reflection Mapping – Paraboloid

- From view point of reflector:
  - Draw two images
  - Transfrom vertices as usual but:
    - Divide X, Y, and Z by W
      - Call the magnitude of this vector L
    - Normalize and divide X and Y by (Z + 1)
    - Set Z to L remapped to view volume
      - Usual [0, 1] mapping based on near / far
    - Set W to 1.0

References

http://opengl.org/resources/code/samples/sig99/advanced99/notes/node184.html

Jason Zink. “Dual Paraboloid Mapping in the Vertex Shader.” GameDev.net, 1996. http://www.gamedev.net/reference/articles/article2308.asp

Wolfgang Heidrich and Hans-Peter Seidel. “View-independent environment maps.” In *Proceedings of the SIGGRAPH/Eurographics Worksjhop on Graphics Hardware*, 1998. http://www.cs.ubc.ca/~heidrich/Papers/GH.98.pdf

Michael Ashikhmin and Abhijeet Ghosh. “Simple Blurry Reflections with Environment Maps.” Journal of Graphics Tools, 7(4): 3-8, 2002. http://people.ict.usc.edu/~ghosh/papers.html

R. Ramamoorthi and P. Hanraham. “An Efficient Representation for Irradiance Environment Maps.” In *Proceedings of SIGGRAPH 2001, Computer Graphics Proceedings*, Annual Conference Series, edited by E. Fiume, pp. 497–500, Reading, MA: Addison-Wesley, 2001. http://www-graphics.stanford.edu/papers/envmap/

Reflection Maps as Lights

- Just like reflection mapping:
  - Render the “light” into the reflection map
  - The part of the reflection map that isn't the light is black
    - Can put multiple lights in one reflection map

Reflection Maps as Lights

- What is the limitation of this simple approach?

Reflection Maps as Lights

- What is the limitation of this simple approach?
  - Really only works for perfectly mirror-like surfaces
    - Surfaces where the specular exponent approaches ∞
  - Essentially creates an aliasing problem
    - Only one sample is taken from the environment

Reflection Maps as Lights

- If under-sampling is the problem, how can we fix it?

Reflection Maps as Lights

- If under-sampling is the problem, how can we fix it?
  - Obvious answer: take more samples
  - Filter the samples together
    - The lighting equation supplies the sample weights

Reflection Maps as Lights

- What is the problem with this technique?

Reflection Maps as Lights

- What is the problem with this technique?
  - Taking enough samples to get good results is slow
  - Taking few enough samples to be fast gives poor results
- *Remind* you of anything?
  - And what was the solution there?

Reflection Maps as Lights

- Just like texture minification!
  - The answer there was to create pre-filtered versions of the texture called mipmaps
- Create new reflection maps:
  - Each texel in the new map is created from *all* of the texels in the old map filtered using weights from the lighting equation
  - This is expensive, but it only has to be done once... and that can be off-line

Reflection Maps as Lights

- Notes / caveats:
  - The new reflection map only includes the specular component
  - Must be generated with a constant V, so the resulting reflection map is view-dependent
  - Can create a second map for diffuse lighting
    - Use the diffuse lighting equation
    - Use the surface normal instead of the reflection vector
    - This type of reflection map is called an *irradiance map*

Fresnel Reflection

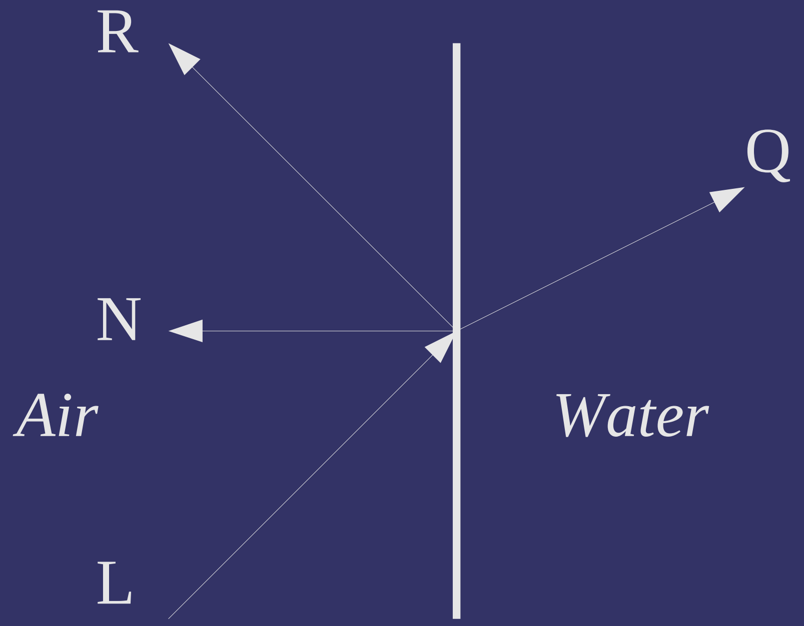
- Named after French physicist Augustin-Jean Fresnel
  - It's French... It's pronounced *fray-NELL*
- Light moves at different speeds through different materials
  - The ratio of the speed of light in a vacuum to the speed in a particular material is the *refractive index* of that material
  - Glass has an index of refraction of ~1.5

Fresnel Reflection

- When light passes between material with differing indicies of refraction:
  - The light changes velocity
    - Speed changes
    - Direction changes
    - Wave theory of light: the change in speed causes the change in direction
  - Some of the light is reflected
  - The remaining light is refracted
    - This light passes into the material

Wave Theory – Refraction

- When light leaves one material and enters another, it changes direction
  - At the *interface* the speed changes, and the light bends



Wave Theory – Refraction

Image from http://en.wikipedia.org/wiki/File:Refraction-with-soda-straw.jpg

Reflection vs. Refraction

- Ratio of reflection to refraction depends on the angle between the light and the normal at the interface
  - The larger the angle between the normal and the light, the more light is reflected
  - The effect is like a rock skipping on water
    - The greater the angle between the rock's velocity and the water's surface normal, the more skipping

Reflection Math

- The amount of reflection $R(\theta)$ is:

$$c = n_i / n_t (\cos\theta)$$

$$g = \sqrt{1 + c^2 - (n_i/n_t)^2}$$

$$R(\theta) = \frac{1}{2}\left(\frac{(g-c)}{(g+c)}\right)^2\left(1+\left(\frac{c(g+c)-(n_i/n_t)^2}{c(g-c)+(n_i/n_t)^2}\right)^2\right)$$

  - n_i is the refractive index of the first material
  - n_t is the refractive index of the second material
  - θ is the angle between the surface normal and the light vector

Reflection Math

- Yewouch! That math is complex and expensive
- A good approximation exists:

$$R_a(\theta) = R(0) + (1 - R(0))(1 - \cos(\theta))^5$$

  - $R(0)$ is calculated in the application and passed into the shader as a uniform

Fresnel Reflection in Lighting

- Simulate a diffuse surface with a shinny coating:

$$K = (1 - F) K_d + F K_s$$

  - The Fresnel term determines what part of the light is reflected by the specular coating
  - The light that isn't reflected by the specular coating is reflected by the diffuse layer

Fresnel Reflection and Materials

- Dielectric materials exhibit a strong Fresnel factor
  - Dielectric means that it does *not* conduct electricity
  - Plastic, glass, automotive paint, etc. are dielectic and have strong Fresnel factors
  - Metal is a conductor and has almost no Fresnel factor
    - This fact will be very important later...

References

Wloka, Matthias, Fresnel Reflection. NVIDIA. July 2002. http://developer.nvidia.com/object/fresnel_wp.html

Westin, Stephen. “Fresnel Reflectance.” September 2007. http://www.graphics.cornell.edu/~westin/misc/fresnel.html

“Reflection and Refraction of Light (Fresnel Formulas).” http://physics-animations.com/Physics/English/rays_txt.htm

http://en.wikipedia.org/wiki/Fresnel_equations

Reading for Next Week

Cook, Robert L. and Torrance, Kenneth E., "A Reflectance Model for Computer Graphics." In *SIGGRAPH ’81: Proceedings of the 8th Annual Conference on Computer Graphics and Interactive Techniques*, pages 307–316. ACM, 1981. http://graphics.pixar.com/library/ReflectanceModel/

Next week...

- Quiz #1
- Assignment #1 due
- BRDFs, part 1
  - Common ideas and terminology
  - Cook-Torrance BRDF
  - Micro-facet based BRDFs

Legal Statement

This work represents the view of the authors and does not necessarily represent the view of Intel 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.