Computer Graphics Programming I

- Agenda:
  - Quiz #3
  - Per-fragment lighting
    - Dot3 texture combiner for lighting
    - Tangent space lighting
    - Bump mapping
  - Assignment #3 due
  - Start assignment #4

Phong Shading Recap

- Interpolate normals between vertices
  - If polygons are large, we will probably need to re-normalize the interpolated values.
- Interpolate *H* vector between vertices
  - Again with the re-normalize step
- Perform $(N \cdot H)^n$ per-fragment.

Phong Shading in Texture Combiners

- The *N*•*H* calculation is the easy part.
  - Use `GL_DOT3_RGB`.
- What are the hard parts?

Phong Shading in Texture Combiners

- The *N*•*H* calculation is the easy part.
  - Use `GL_DOT3_RGB`.
- What are the hard parts?
  - Where does *N* come from?
  - Where does *H* come from?
  - What about diffuse lighting?
  - Specular exponent.

Surface-Space

- From the point of view of the surface (i.e., in *surface-space*), what is the normal vector?

Surface-Space

- From the point of view of the surface (i.e., in *surface-space*), what is the normal vector?
  - Assuming the surface is flat, N = (0, 0, 1).

Surface-Space

- From the point of view of the surface (i.e., in *surface-space*), what is the normal vector?
  - Assuming the surface is flat, N = (0, 0, 1).
- If we know the world-space surface normal, N_{surf}, can we create a transformation that will map N_{surf} to (0, 0, 1)?

Surface-Space

- From the point of view of the surface (i.e., in *surface-space*), what is the normal vector?
  - Assuming the surface is flat, N = (0, 0, 1).
- If we know the world-space surface normal, N_{surf}, can we create a transformation that will map N_{surf} to (0, 0, 1)?
  - Not uniquely.
  - If we knew another vector in the plane, we could create this transformation.

Tangents

- Call this new vector the *tangent vector*, and note it T_{surf}.
  - Knowing N_{surf} and T_{surf} is enough the create an orthonormal basis.
  - This basis can transform any vector into surface-space.
  - Tangent vectors can be created automatically (tricky) or by hand (annoying).

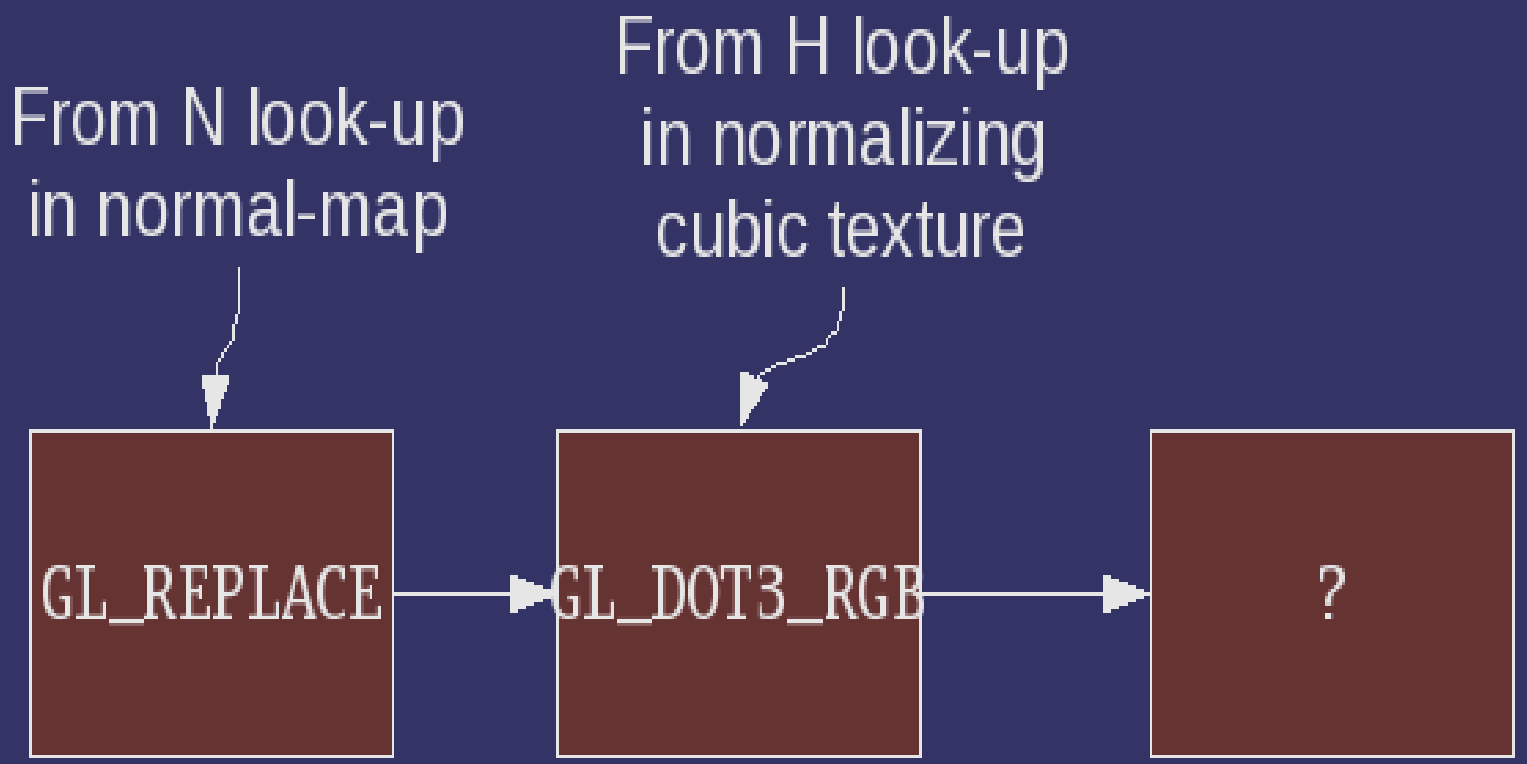
Where does H come from?

- Calculate the surface-space transformation.
- Calculate *H* per-vertex.
  - We covered this calculation in week 3.
- Transform the per-vertex *H* vector to surface space.
- Use the *H* vector as a texture coordinate.
  - This will perform the interpolation.
- Use a cubic texture to re-normalize *H*.

Where does N come from?

- Three ways to get *N*:
  - If surface is flat: N is constant (0, 0, 1), store in a combiner constant color.
  - If surface is curved: store per-vertex normal in one of the interpolated colors.
  - Surface is bumpy: fetch *N* from a texture.
    - Texture is stored so that R, G, and B map to the X, Y, and Z of the normal in surface space.
    - These textures tend to look blue because the Z component is usually close to 1.0.

Combiner Setup



What about the exponent?

- Without shaders, we're very limited.
  - Can burn a texture unit and do (GL_PREVIOUS, GL_PREVIOUS, GL_MODULATE) to square it.
  - Can do multiple passes to generate higher exponents.

What about diffuse?

- If there are more texture units, use them to do diffuse calculation.
  - *L* vector needs same treatment for diffuse as *H*.
- Otherwise, do diffuse as a separate pass.
  - We'll cover multi-pass next week.
- If the hardware has crossbar-like functionality, we can use one less texture stage for the specular calculation.

Next week...

- Fog
- Framebuffer operations
  - Blending
  - Alpha test
- Multi-pass rendering
- Term projects assigned!!!

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.