CSE4431: Lab 3

Burton Ma

Mon 30 Jan, 2012

1 Introduction

In this lab you will implement simple texture mapping (replace, modulate, and specular preserving), multiple texture mapping, gloss mapping, and bump mapping.

2 Files

Copy the files from the directory `/cse/course/4431/labs/03/` into a directory owned by you.

3 Basic Image Texture Mapping

Open the `earth.frag` fragment shader in your favorite editor. The fragment shader uses a `uniform sampler2D` variable. Samplers are special types that are used to acess texture values. This particular shader uses the sampler to retrieve the daytime color of a point on the earth from a texture and sets the fragment color to the texture color. This is similar to the OpenGL decal texture mode.

Open the `earth.vert` vertex shader in your favorite editor. This is essentially the ADS vertex shader from the previous lab. To perform texture mapping, we need to assign texture coordinates to each vertex. `glman` assigns texture coordinates for its built-in models, but we will compute our own coordinates for this lab.

The function `ComputeTextureCoords` should compute the texture coordinates for a given point on a sphere of radius 1 centered on the origin. The standard parameterization of a sphere is given by

$$(x,\ y,\ z) = (\cos\theta\sin\phi, \sin\theta\sin\phi, \cos\phi), \quad 0 \le \phi < 2\pi,\ 0 \le \phi \le \pi \tag{1}$$

Complete `ComputeTextureCoords` so that it returns a `vec2` with coordinates (s, t) such that s is θ and t is ϕ, both scaled to lie between 0 and 1.

Once complete, load `earth.glib` in `glman` and observe the results. You will have to modify your calculation of s to get a correct image of Earth.

4 Modulate Lighting

The fragment shader in the previous section uses the texture color for the fragment color; this is essentially the way decal mode behaves in OpenGL. Modify the shader so that it assigns `vColor` to the fragment color and observe that there is a prominent specular highlight. Now, modify the shader so that implements OpenGL's modulate texture mode (multiplies the texture and lighting colors). Observe how strongly the specular highlight is attenuated.

5 Preserve Specular Highlights

Notice that the vertex shader outputs the diffuse and specular intensities (the values of $N \cdot V$ and $(V \cdot R)^f$, respectively). Modify the fragment shader so that it takes the variables `diffuseIntensity` and `specularIntensity` as inputs. Compute the fragment color as the texture color multiplied by the diffuse intensity and add the specular intensity (perhaps scaled by some value between 0 and 1). Observe how this modification preserves the specular highlight while modulating the texture color by the diffuse intensity.

6 Combining Two Textures

Open the fragment shader `earth_daynight.frag`; this shader uses two textures, one of the earth during the day and the other of the earth during the night. Complete the shader so that it:

- computes the daytime color as the daytime texture color modulated by the diffuse intensity
- computes the nighttime color as the nighttime texture color
- computes the fragment color as a blend of the daytime and nighttime colors. If the diffuse intensity is greater than 0.2 you should assume that the fragment location is in daylight. If the diffuse intensity is zero then you should assume that the fragment location is in nighttime. If the diffuse intensity is between 0 and 0.2 you should blend the daytime and nighttime colors using the `mix` function (see textbook page 104, or look it up online).

7 Gloss Mapping

A gloss map is a texture that describes the variation in shininess on a surface. The gloss map value modulates the specular light component. If the gloss map value for a point is 1 then the point is shiny (i.e., should include a specular component in the lighting calculation); if the gloss map is 0 then the point is not shiny (i.e., the specular component should be set to zero). For intermediate gloss values, the specular component should be multiplied by the gloss value.

When rendering the earth, the land should not be shiny and the water should be shiny. The gloss map `earth_gloss.bmp` is white everywhere there is water and black everywhere there is land.

Open the fragment shader `earth_daynightgloss.frag`; this shader uses three textures, one of the earth during the day, one of the other of the earth during the night, and a gloss map. Complete the shader so that it is identical to the previous shader except that the daytime color now includes the specular intensity modulated by the gloss map. Notice that when a specular highlight overlaps both land and water only the part of the highlight over water appears.

8 Bump Mapping

Read the handout on bump mapping that accompanies this lab, and then modify the vertex shader `bump.vert` to perform the following steps:

1. Compute the view direction in eye coordinates (the vector from the vertex position to the camera position)
2. Compute the light direction in eye coordinates (the vector from the vertex position to the light position)
3. Compute the vector $n = T_{\text{normal}} * n_{\text{model}}$ where T_{normal} is the normal transformation matrix and n_{model} is the normal vector in model coordinates. Make sure to normalize the vector.
4. Compute the vector $t = T_{\text{modelview}} * t_{\text{model}}$ where $T_{\text{modelview}}$ is the modelview transformation matrix and t_{model} is the tangent vector in model coordinates. You do not need to normalize this vector, as it will be normalized in Step 6.
5. Compute the vector $b = t \times n$ where $\times$ is the cross product. Make sure to normalize the vector.
6. Recompute the vector $t = n \times b$; this step is not normally necessary, but the tangent vector supplied by `glman` appears to be incorrect. Make sure to normalize the vector.
7. Form the 3×3 matrix $BTN = \begin{bmatrix} b_x & b_y & b_z \\ t_x & t_y & t_z \\ n_x & n_y & n_z \end{bmatrix}$
8. Use BTN to transform the view direction computed in Step 1; the result should be stored in the out variable `vViewDir`
9. Use BTN to transform the light direction computed in Step 2; the result should be stored in the out variable `vLightDir`

Load the file `bump.glib` in `glman`. If you have done all nine steps correctly then you should see a smooth sphere with a specular highlight. Edit the fragment shader `bump.frag` and change the value of `f` to 0.2 in the `main` method. The sphere should now appear to have a bumpy surface; larger values of `f` will increase the bumpiness.

The fragment shader works by interpreting the red and green components of the bump map as displacements in the b and t directions of the normal vector for each fragment location. Because the texture values lie between 0 and 1 we need to subtract 0.5 from the texture values to get displacements in the positive and negative directions. In the local surface coordinate system, the original normal vector has a direction of $(0,\ 0,\ 1)$; the bump map simply replaces the first two components of the original normal vector to obtain a displaced normal vector, which is then passed to an ADS shader (along with the light and view directions in the local surface coordinate frame).