CS 465 Homework 8

out: Friday 27 October 2006
due: Friday 3 November 2006

This problem considers modifications and extensions to the standard pipeline architecture presented in class. Suppose you wanted to implement Phong lighting (with a fixed exponent p) as an option in the fragment processor. You could do this with a fragment program like this:

shade-fragment($\mathbf{n}$, k_d, k_s, p, I, $\mathbf{v}_L$, $\mathbf{v}_E$)
  $\mathbf{v}_H = \text{normalize}(\mathbf{v}_L + \mathbf{v}_E)$
  $r_d = k_d \max(0, \mathbf{v}_L \cdot \mathbf{n})$
  $r_s = k_s \max(0, (\mathbf{v}_H \cdot \mathbf{n})^p)$
  return $I(r_d + r_s)$

The meanings of the symbols are as defined in the "lighting and shading I" lecture notes. Our surface has a spatially varying diffuse color, but the specular parameters are uniform over the surface. We are using a local viewer and a directional light.

Part A

1. The fragment program has several input arguments. Which of these need to be interpolated by the rasterizer and which are constants?

Part B

In order to speed up processing, we introduce the following optimization: create a 16-bit grayscale texture map T that is of size 256×1, and set $T(i, 0) = \text{round}((2^{16} - 1)(\frac{i}{255})^p)$. Now in our fragment processor, instead of computing $(\mathbf{v}_H \cdot \mathbf{n})^p$, we compute:

$i = 255 \max(0, \mathbf{v}_H \cdot \mathbf{n})$
$r_s = k_s$ fetch-texture$(T, i, 0)/(2^{16} - 1)$

That is, we calculate an index into our texture map and retrieve the stored value at that index using the hardware function fetch-texture, which accepts the texture variable and row

and column indices and returns the value of the texture at that position. If either of the row or column indices are not integers, the hardware returns the bilinearly interpolated texture value from the nearest integer indices.

Now we wish to modify our shader so that the specular coefficient k_s and the Phong exponent p can vary smoothly across the surface, where $p \in [0, 255]$.

1. Describe how to implement this using a texture map T of size 256×256
2. Give pseudocode for the fragment program in a form similar to above.
3. Which parameters are varying and which are constant?
4. What is the value of $T(140, 110)$?
5. What value will your fragment program return for $\mathbf{n} = \frac{1}{\sqrt{41}}[1; 6; 2]$, $\mathbf{v}_E = \frac{1}{5}[0; 3; 4]$, $\mathbf{v}_L = \frac{1}{\sqrt{17}}[1; 0; 4]$, $p = 3.6$, $k_s = 0.8$, $k_d = 0.2$, $I = 1$
6. What is the relative error of your answer from part (5) compared to the true value?

Part C

Now we wish to implement a more advanced reflection model, the Ward model. In this model the reflected color is defined by the equation:

$$r_s = k_s \sqrt{\frac{\cos\theta_E}{\cos\theta_L}} \frac{e^{-\left(\frac{\tan\alpha}{m}\right)^2}}{m^2}$$

where θ_E is the angle between $\mathbf{v}_E$ and $\mathbf{n}$, θ_L is the angle between $\mathbf{v}_L$ and $\mathbf{n}$, α is the angle between $\mathbf{v}_H$ and $\mathbf{n}$, and $m \in [0, 1]$ is the parameter that controls the size of the highlight.

1. Describe how this model can be implemented by a similar approach using texture maps. Use the simplest texture maps possible (that is, use 1D maps in preference to 2D maps and fewer maps in preference to more maps) while avoiding the computation of exponentials, square roots, and inverse trigonometric functions in the shader.
2. Give pseudocode for the fragment program in a form similar to above.
3. Which parameters are varying and which are constant?