Computer Graphics
CSE 591, Spring 1998

David Salesin, 418 Sieg
Head TA: Doug Zongker

Week 5 study sheet: Shading

Problem 1 (20 points) In the Phong illumination model, specular reflections from each light source i are approximated by taking the dot product between the direction of the viewer $\widehat{v}$ and the direction of the reflected light $\widehat{r}_i$, and raising the result to some power x.

a) (6 points) Derive an expression for the direction of the reflected light $\widehat{r}_i$ in terms of the directions of the light source $\widehat{\ell}_i$ and surface normal $\widehat{n}$.

b) (4 points) Often, in shading computations, the viewer direction $\widehat{v}$ and light source direction $\widehat{\ell}_i$ are assumed to be constants. Under what circumstances might such an assumption be justified?

Problem 1 (continued)

Blinn and Newell have suggested that, when $\widehat{v}$ and $\widehat{\ell}_i$ are assumed to be constants, the computation of $\widehat{v} \cdot \widehat{r}_i$ can be simplified by associating with each light source i a fictitious light source that will generate specular reflections. This second light source is located in a direction $\widehat{h}_i$ halfway between $\widehat{\ell}_i$ and $\widehat{v}$. The specular component is then computed from $(\widehat{n} \cdot \widehat{h}_i)^x$ instead of from $(\widehat{v} \cdot \widehat{r}_i)^x$.

c) (3 points) How does this new equation simplify the shading computation?

d) (7 points) In what way might the new equation change the appearance of the specular highlights? Justify your answer.

Problem 2 (18 points) Texture mapping has many applications in computer graphics.

a) (5 points) List 5 different applications of texture mapping, including at least one that has nothing to do with simulating the appearance of 3D objects.

The computationally difficult part of texture mapping is in summing over all of the pixels covered by a particular quadrilateral in the $r \times r$-pixel texture map.

In class, we discussed four different ways that this process can be implemented:

1. The "brute force" method
2. Mip maps
3. Summed area tables
4. Stochastic sampling

b) (4 points) Suppose we're using texture mapping to determine the color of a single pixel P in screen space, computed by averaging all of the n pixels in the texture map covered by the image of P in texture space. For each of the four methods above, what is the asymptotic time complexity of computing this average (in terms of n and r)?

Problem 2 (continued)

c) (2 points) Suppose that the pixels in the original texture map are each represented with b bits. How many bits are there in the original texture map?

In order to allow for faster texture mapping, some of the methods on the previous page use a preprocessing step that transforms the original texture map T into a new structure T', which is then used instead of T in the texture-mapping process itself. The new structure T' may sometimes require additional storage.

Let's define the "storage overhead" of a method to be the ratio of the size of T' to the size of T. (If a method requires no preprocessing, then we'll assume $T' = T$ and define the storage overhead to be 1.)

d) (7 points) What is the storage overhead of each of the four methods on the previous page? (Hint: Don't forget to consider the number of bits required to accurately represent the sum of two b-bit integers.)