Computer Graphics	Prof. Brian Curless
CSE 457	Autumn 2000

Homework #2

Shading, ray tracing, texture mapping

Prepared by: Aaron Eakin, Doug Johnson, and Brian Curless

Assigned: Monday, November 13th
Due: Wednesday, November 22th

Directions: Please provide short written answers to the questions in the space provided. If you require extra space, you may staple additional pages to the back of your assignment. Feel free to talk over the problems with classmates, but please *answer the questions on your own*.

Name:_____

Problem 1. Phong shading

The Phong shading model for a scene illuminated by a global ambient light and a single directional light can be summarized by the following equation:

$$I_{phong} = k_e + k_a I_a + I_l[k_d(\mathbf{N}\cdot\mathbf{L})_+ + k_s(\mathbf{V}\cdot\mathbf{R})_+^{ns}]$$

Imagine a scene consisting of a sphere illuminated by white global ambient light and a single white directional light. Assume the directional light is pointing in the same direction as the viewer. Describe the effect of the following conditions on the shading of the object. At each incremental step, assume that all the preceding steps have been applied first.

a. The directional light is off. How does the shading vary over the surface of the object?

b. Now turn the directional light on. The specular reflection coefficient k_s and the specular exponent n_s of the material are both zero, and the diffuse reflection coefficient k_d is (0.5,0.5,0.5). How does the shading vary over the surface of the object?

c. Now set the specular exponent, n_s, to 20. How does the shading change?

d. Now rotate the sphere about its own origin and then translate it straight towards the viewer. What happens to the shading of the sphere?

e. Now increase the specular reflection coefficient k_s of the material to be (0.5,0.5,0.5). What happens?

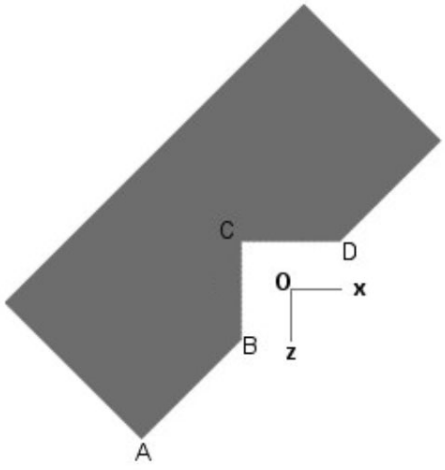
f. Imagine that the diffuse reflection coefficient kd is actually (0.5,0.5,0) and the specular coefficient ks is actually (0,0,0.5). How does the *colo*r shading of the sphere vary?

g. Now increase the specular exponent n_s. What happens?

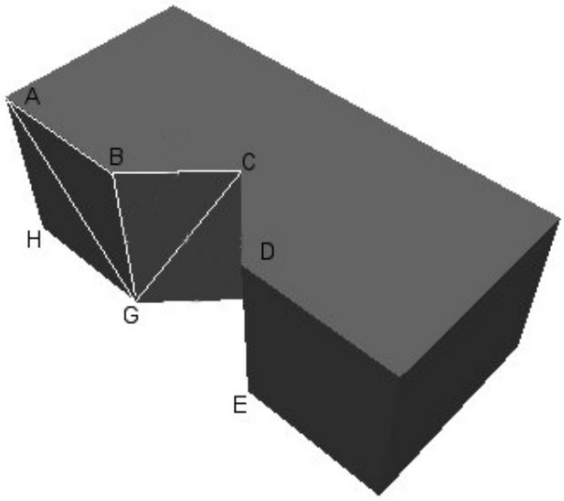
Problem 2. Interpolated Shading

The figure below shows a block with a triangular groove in it. The block is sitting flat on the x-z plane (y=0), and extends upwards to y=1. Looking down from the top, the groove is defined by the three points B = (-1,1,1), C=(-1,1,-1), and D=(1,1,-1), and extends straight down the y-axis. The viewer is at (10,0.5,10) looking straight into the groove. There is a directional light shining on the scene along the vector (-1,0,-1).

Overhead (plan) view



Full view



a. In order to shade the front face of the block to be flat in OpenGL you need to specify the normal at each vertex as perpendicular to the face. What normal vector would you specify at each vertex of the triangle GBA while drawing the front face of the block?

Triangle	Vertex	Normal
GBA	G	
GBA	B	
GBA	A	

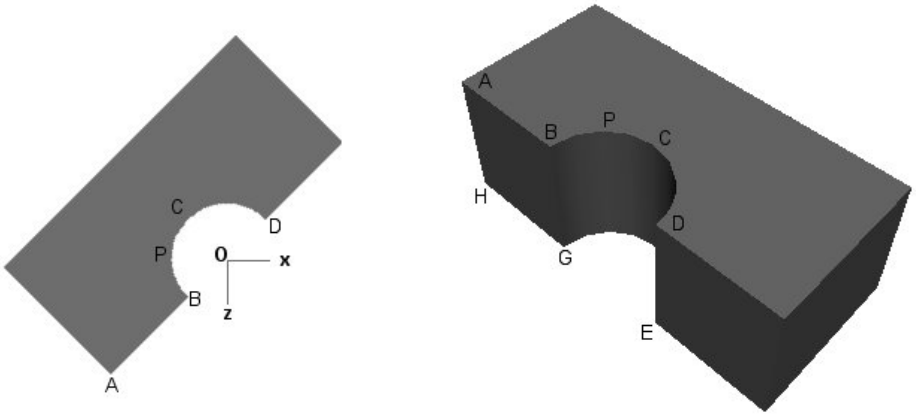
b. In order to draw the faces of the groove as flat shaded triangles you need to specify the normal at each vertex as perpendicular to the face of the groove. In this case, what normal vector would you specify at each vertex for the triangle GCB?

Triangle	Vertex	Normal
GCB	G	
GCB	C	
GCB	B	

c. Note that the normal vector that you specified for vertex B may be different in parts (a) and (b). If they were different, would this be a problem for OpenGL?

d. If the vertices of triangle GCB are drawn with OpenGL in GL_TRIANGLES mode in the order G, then C, then B, is the viewer looking at the front or the back of the triangle?

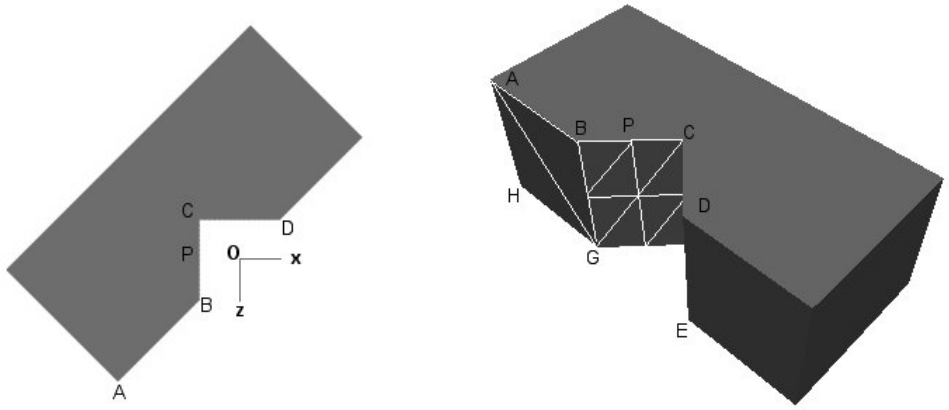
e. Let's now assume that this square groove is actually a crude approximation to a half-cylinder groove. If you want the shading of the groove to give the appearance of being rounded off inside instead of the sharp corner that the geometry shows (ie, half of a round cylinder instead of half of a "square cylinder"), how would you specify the unit normals at the vertices when drawing GCB? (See figure. At this point, we are trying to simulate this look with shading only, not geometry changes. Note that point P is added for reference in part (g).)



Triangle	Vertex	Normal
GCB	G	
GCB	C	
GCB	B	

f. Assume that the normals have been set as in (e), and that there are non-zero diffuse and specular components to the light and material values. Now switch from OpenGL's Gouraud interpolated shading to Phong interpolated shading. How will the diffuse appearance of the groove change (if at all)? How will the specular appearance of the groove change (if at all)?

g. If we're trying to better approximate the geometry of the cylindrical groove, one easy-to-calculate improvement is to subdivide each triangular face inside the groove by introduce midpoints along the edges of the triangle followed by introducing four new as shown. If you subdivided the groove face that triangle GCB is on, as shown in the figure below (compare with (a)), what would be the best choice for the new coordinates and the normal of new point P? Why? For reference, point P is illustrated in part (e).

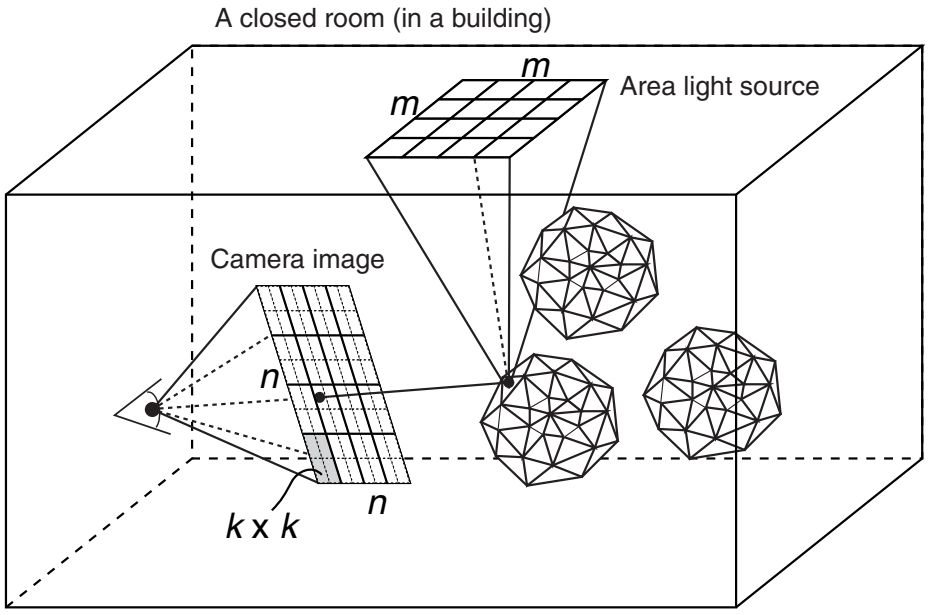


Vertex	Coordinates	Normal
P		

h. How would you continue to improve the approximation to the rounded groove?

Problem 3. Ray Tracing (Z-Buffer and Distribution Ray Tracing)

Suppose we want to combine the Z-buffer algorithm with ray tracing (assuming that the scene consists only of polygons). We can assign to each polygon a unique emissive color corresponding to an "object ID". Then, we turn off all lighting and render the scene from a given point of view. At each pixel, the Z-buffer now contains the object point (indicated by the pixel x,y coords and the z-value stored in the buffer) and an object ID which we can look up to figure out the shading parameters. In effect, we have done a ray cast for a set of rays passing through a given point (the center or projection). The following figure will be a reference for the terms used in the rest of this problem:



a. The ray cast from the eye point through an n x n image (projection) plane with k x k supersampling is actually going to be computed using the Z-buffer algorithm described above. In effect , how many rays are cast from the eye to determine the first intersected surfaces?

b. Suppose we now want to compute the contribution from an area light source in order to get soft shoadows. Consider the ray from the eye that hits a surface as shown. We can now set up an image plane over the light source with resolution m x m and render the scene with Z-buffers and object ID's. Where would you put the far clipping plane and how would you use the resulting image as a step in the process of computing soft shadows?

c. Let's say we follow this procedure for all pixels for a depth of q bounces in a scene assuming non-transmissive surfaces. In effect, how many rays (including the eye rays) will we end up tracing?

d. Now, instead of using the z-buffer algorithm, we use the distribution ray tracing method to achieve soft shadows. As above, we use k x k supersampling in the image plane, though in this case the rays are jittered. How many rays (including the eye rays) do we need to trace for a depth of q bounces in a scene assuming non-transmissive surfaces?

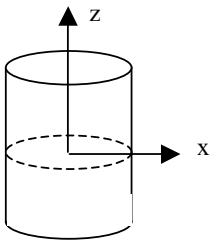
d. Given your answers to (b) and (c), when would it be appropriate to use the modified Z-buffer in a ray tracing algorithm with anti-aliasing and soft shadows?

Problem 4. Procedural Solid Textures

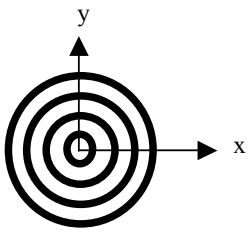
One problem with the standard method of texture mapping is that textures appear to be stuck on objects in the same manner that wallpaper is put on a wall. In many cases, a surface doesn't look like it is really made up of the material that is texture mapped onto it, for example, consider a wood texture map. One method that is used to alleviate the "wood veneer" problem is to use a solid, or procedural, texture map.

A procedural texture map answers the question: What is the color of the surface at this point?

Here is an example of a horizontal slice of a cylinder with our wood shader applied to it:



Original Cylinder



Horizontal slice

a. Fill out the following function with very general pseudocode. Your code will return the answer to the question: What is the color of the wood at this point on the surface? Assume that you have values for the age of the wood, and the two colors for each type of ring in: age, darkRing, and lightRing. The Point passed into the function contains the x, y, and z coordinates of the point on the surface in object space.

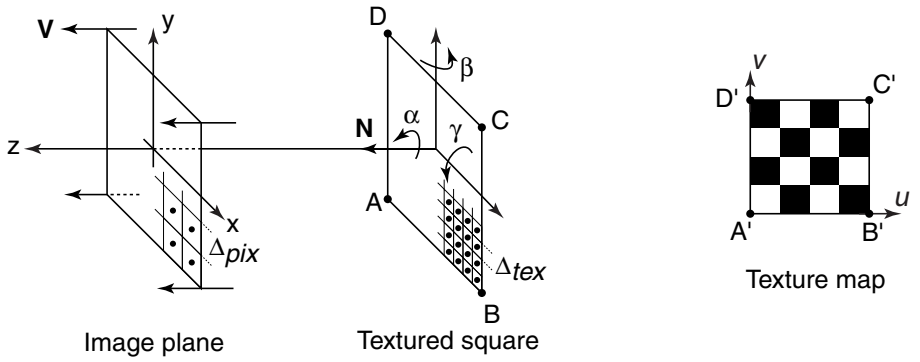
```
color wood (Point pt) {




}
```

b. As the illustration above shows, this pattern is somewhat boring. It looks too perfect. How would you modify the code to introduce a less ‘sterile’ appearance, specifically, is there a way to introduce some ‘wiggles’ into the rings?

c. Let’s say that during the seasons that yield light rings, the tree was absorbing silver from some strange atmospheric conditions. How would you modify the above function so that light colored rings were shinier than dark colored rings?

Problem 5. Texture filtering (10 points)

In class, we discussed how brute forces sampling, mip maps, and summed area tables can be employed to anti-alias textures. The latter two techniques average over a region of the texture image very quickly with varying degrees of accuracy, which we consider further in this problem. Consider the scene below: an orthographic viewer looking down the –z-axis views a textured square. The image size and square size are the same and they are initially aligned to one another as shown. The pixel spacing on the image plane and the texel spacing on the square are Δ_{pix} and Δ_{tex}, respectively.



a. Assuming $\Delta_{pix} > \Delta_{tex}$, how must these sample spacings be related in order for mip mapping to yield the correct values without interpolating among mip map levels?

b. Consider the coordinate system of the square shown in terms of the normal **N** and the two axes aligned with the x and y axes in the figure. Assume that we have the freedom to rotate the square about any *one* of those local axes, as indicated by rotation angles α, β, and γ. What restriction do we have on rotation about any one of these axes in order for mip mapping to return the correct average texture values? [For example, you could decide that α, β, and γ must all be zero degrees, or you could decide that some of them can vary freely, or you can decide that some can take on a set of specific values. Do not focus on rotations that cause the square to be back-facing.]

Problem 5 (cont'd)

c. Now assume we start again with the unrotated geometry and that we're using summed area tables. If linear interpolation within the summed area table causes no significant degradation, what restriction, if any, should we place on the relative pixel and texel spacings to get correct texture averaging?

d. As in (c), what restriction do must we place on rotation about any one of the given axes in order for summed area tables to return average texture values?