Computer Graphics	Prof. Doug Johnson
CSE 457	Autumn 2003

Homework #2

Shading, ray tracing, texture mapping

Prepared by: Mike Beal, Ethel Evans, and Doug Johnson

Assigned: Friday, November 14th
Due: Wednesday, November 26th

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 (12 points)

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?

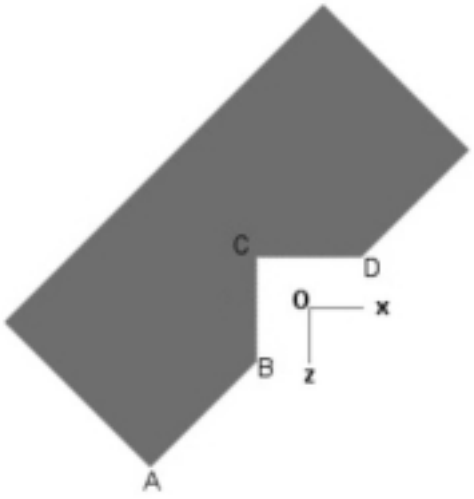
e. Now increase the specular reflection coefficient k_s of the material to be (0.5,0.5,0.5). What happens?

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

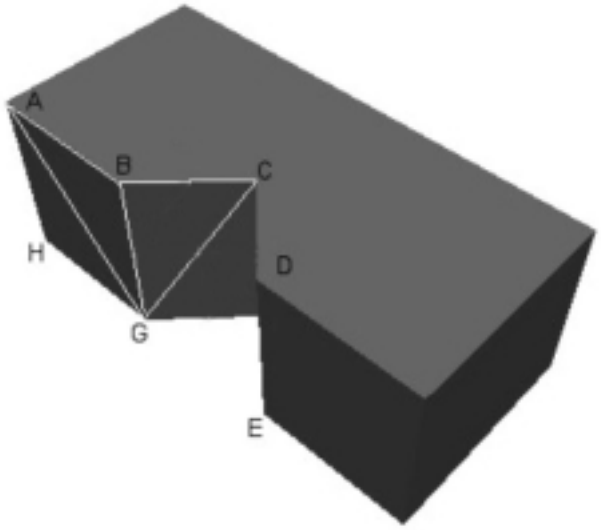
Problem 2. Interpolated Shading (17 points)

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



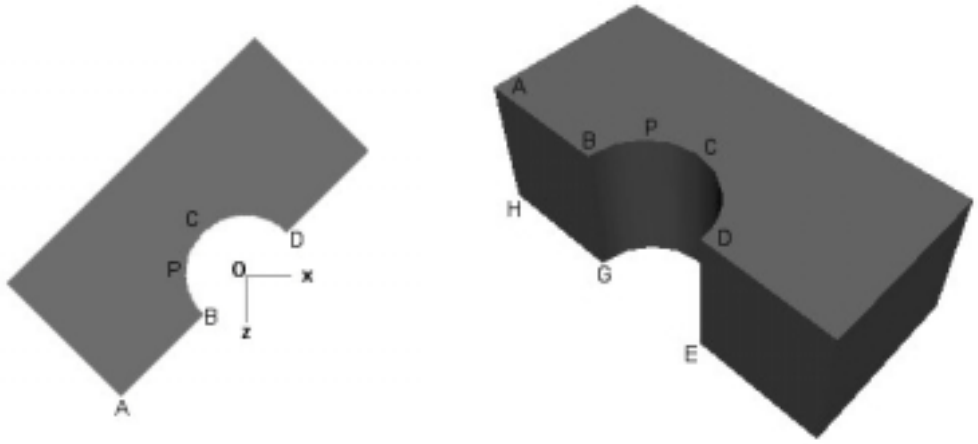
a. In order to shade the front face of the block to be flat in OpenGL you need to specify the normal at each vertex as perpendicular to the face. What normal vector would you specify at each vertex of the triangle GBA while drawing the front face of the block?

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	

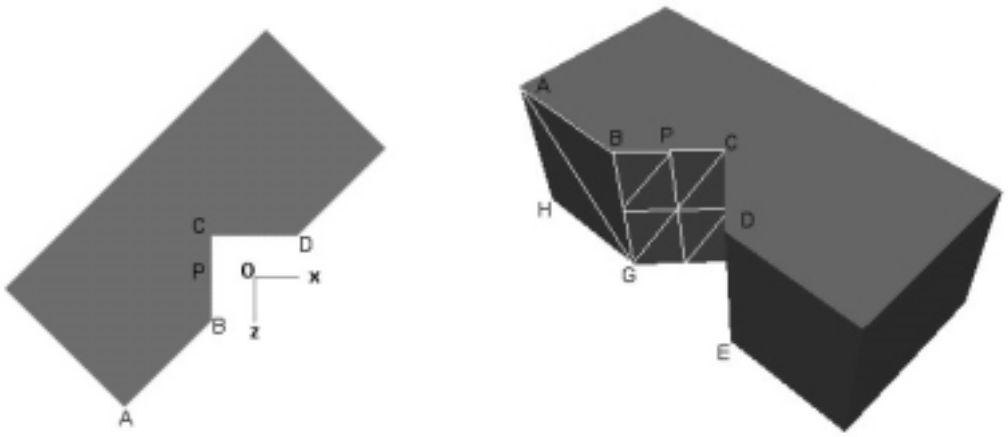
c. Let's now assume that this square groove is actually a crude approximation to a half-cylinder groove. If you want the shading of the groove to give the appearance of being rounded off inside instead of the sharp corner that the geometry shows (ie, half of a round cylinder instead of half of a "square 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	

d. Assume that the normals have been set as in (c), 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)?

e. 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 (c).

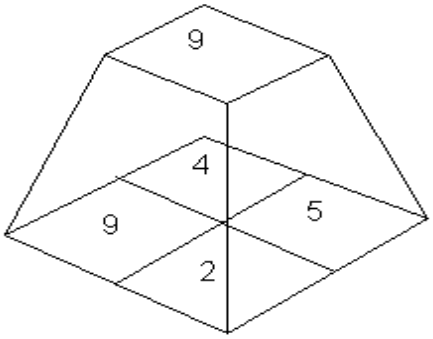


Vertex	Coordinates	Normal
P		

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

Problem 3 Z-buffer (11 points)

The Z-buffer algorithm can be improved by using an image space "Z-pyramid." The basic idea of the Z-pyramid is to use the original Z-buffer as the finest level in the pyramid, and then combine four Z-values at each level into one Z-value at the next coarser level by choosing the farthest (largest) Z from the observer. Every entry in the pyramid therefore represents the farthest (largest) Z for a square area of the Z-buffer. A Z-pyramid for a single 2x2 image is shown below:



a. At the coarsest level of the pyramid there is just a single Z value. What does that Z value represent? Explain briefly.

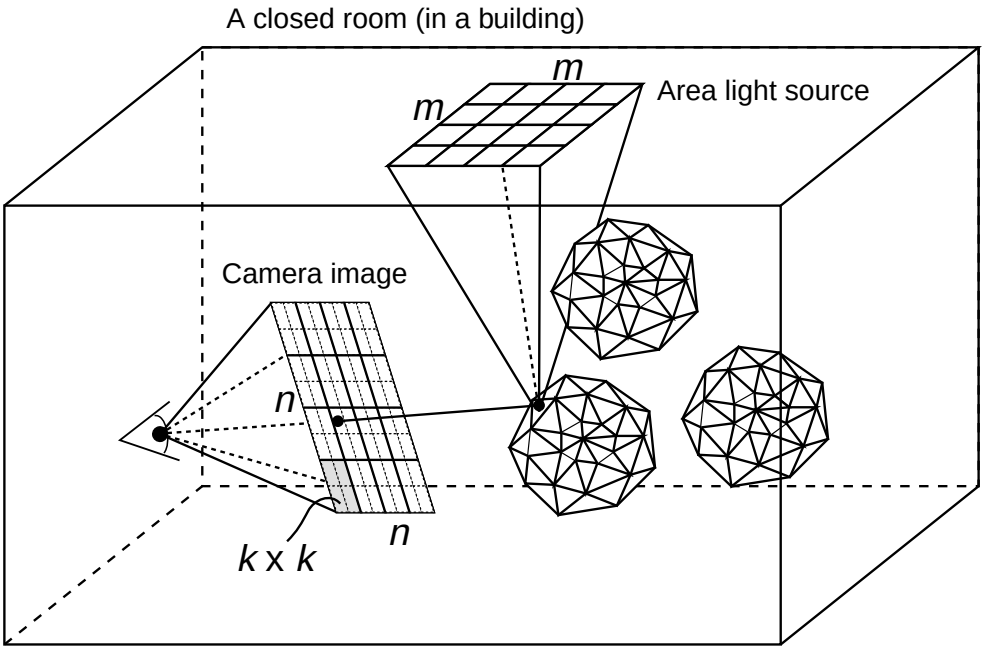
Suppose we wish to test the visibility of a polygon **P**. Let **Zp** be the nearest (smallest) Z value of polygon **P**. **R** is a region on the screen that encloses polygon **P**, and is the smallest region of the Z-pyramid that does so. And let **Zr** be the Z value that is associated with region **R** in the Z-pyramid.

b. What can we conclude if **Zr** < **Zp**? Explain.

c. What can we conclude if **Zp** < **Zr**? Explain.

Problem 4 Z-Buffer and Distribution Ray Tracing (7 points)

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* super-sampling (for anti-aliasing) 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?

Problem 5 Texture Filtering (13 points)

In class, we discussed how brute force sampling, mip maps, and summed area tables can be employed to anti-alias textures.

You are given a 4 x 4 texture map with the following RGB values:

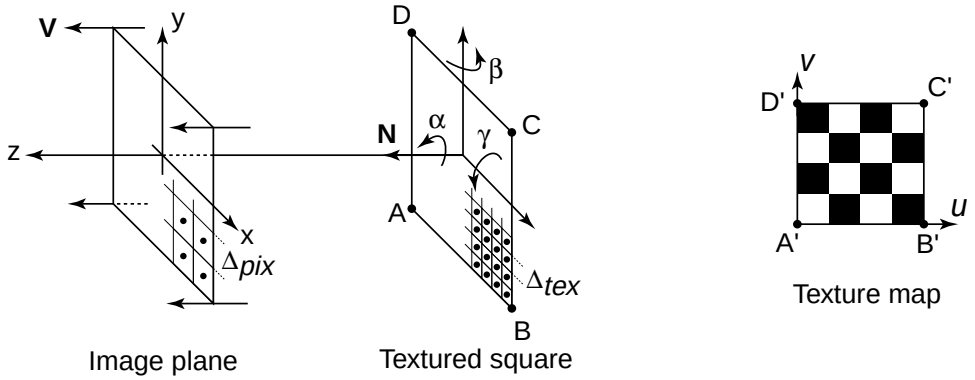
0, 0, 160	0, 40, 120	0, 80, 80	0, 120, 40
40, 0, 120	40, 40, 80	40, 80, 40	40, 120, 0
80, 0, 80	80, 40, 40	80, 80, 0	80, 120, 0
120, 0, 40	120, 40, 0	120, 80, 0	120, 120, 0

Write in the integer values that would be stored into a mip map in the table below.:

Explain your reasoning, briefly.

Problem 6 Texture Filtering (7 points)

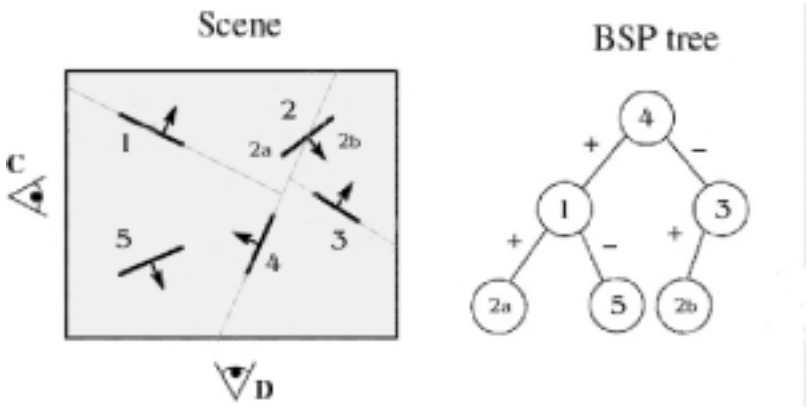
Consider the scene below: an ***orthographic viewer*** looking down the –z-axis views a textured square. The image size and square size are the same and they are initially aligned to one another as shown. The pixel spacing on the image plane and the texel spacing on the square are Δ_{pix} and Δ_{tex}, respectively.



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? Why?

Problem 7. BSP tree (15 points)

Recall that a BSP tree breaks the world up into tree of positive and negative half-spaces that can be traversed to render a scene from an arbitrary viewpoint. Below is the similar to the figure used in class to illustrate the principle of BSP trees:



Recall that each arrow in the scene tells us which way the normal to a polygon is facing and that the normal points to the positive half-space of a polygon.

a. Given viewpoint **C**, list the polygons in the order in which they would be drawn after traversing the BSP tree to give a back to front ordering.

b. Given viewpoint **D**, list the polygons in the order in which they would be drawn after traversing the BSP tree to give a back to front ordering.

c. If we move a polygon in the scene, will we always have to recompute the BSP tree? Justify your answer.