Homework #2

Shading, Ray Tracing, and Texture Mapping

Prepared by: Doug Johnson, Maya Widyasari, and Brian Curless

Assigned: Monday, May 8, 2000
Due: Friday, May 19, 2000

Directions: 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 you own.

Name: _____

Problem 1. Phong shading.

The Phong shading model for a scene illuminated by 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 with one white sphere illuminated by white global ambient light and a single white directional light. 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 of the material is zero, and the diffuse reflection coefficient k_d is non-zero. How does the shading vary over the surface of the object?

c. Now translate the sphere straight towards the viewer. What happens to the shading of the object?

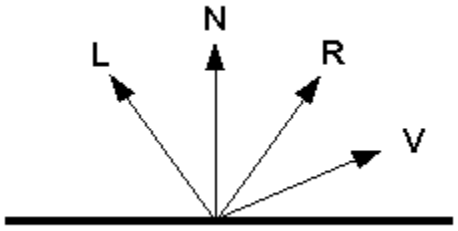
d. Now increase the specular reflection coefficient k_s of the material to be greater than zero. What happens?

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

Problem 2. Halfway vector specular shading

Blinn and Newell have suggested that if **V** and **L** are assumed to be constants the computation of $\mathbf{V} \cdot \mathbf{R}$ in the Phong shading model can be simplified by associating with each light source a fictitious light source that will generate specular reflections. This second light source is located in a direction **H** halfway between **V** and **L**. The specular component is then computed from $(\mathbf{N} \cdot \mathbf{H})^{ns}$, instead of from $(\mathbf{V} \cdot \mathbf{R})^{ns}$.

a. On the diagram below, assume that V and L are the new constant viewing direction and lighting direction vectors. Draw the new direction H on the diagram.

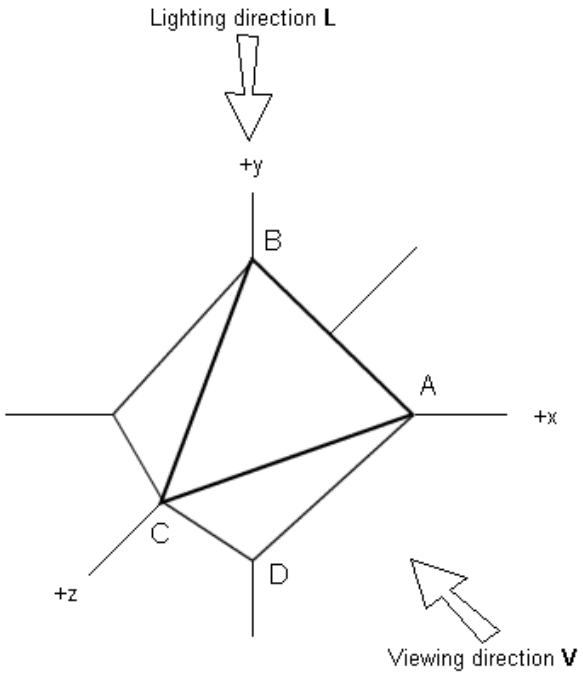


b. Under what circumstances or by making what approximations might **L** and **V** be assumed to be constant?

c. What is an advantage of this approximation? A disadvantage?

Problem 3. Interpolated shading

The faceted polyhedron shown in the figure consists of two pyramids connected at the base (an octahedron) drawn with 8 equilateral triangular faces with vertices at (1,0,0), (0,1,0), (0,0,1), (-1,0,0), (0,-1,0), and (0,0,-1). The viewer center of projection is at (10,0,10) looking directly towards the origin, and there is a directional light shining down from above parallel to the y-axis.



a. In order to draw the faces as flat-shaded triangles in OpenGL, you need to specify the normal at each vertex as perpendicular to the face. What is the unit normal perpendicular to the face at each vertex for the triangles ABC and ACD?

Triangle	Vertex	Coords	Unit Normal
ABC	A	(1,0,0)	
ABC	B	(0,1,0)	
ABC	C	(0,0,1)	
ACD	A	(1,0,0)	
ACD	C	(0,0,1)	
ACD	D	(0,-1,0)	

b. If the vertices of triangle ABC are drawn with OpenGL in GL_TRIANGLES mode in the order A, then B, then C, is the viewer looking at the front or the back face of the triangle?

Problem 3 (cont'd)

c. Assume that this object is really just a crude approximation of a sphere (you are using the octahedron to represent the sphere because your computer is slow). If you want to shade the octahedron so that it approximates the shading of a sphere, what would you specify as the normal at each vertex?

Triangle	Vertex	Coords	Unit Normal
ABC	A	(1,0,0)	
ABC	B	(0,1,0)	
ABC	C	(0,0,1)	
ACD	A	(1,0,0)	
ACD	C	(0,0,1)	
ACD	D	(0,-1,0)	

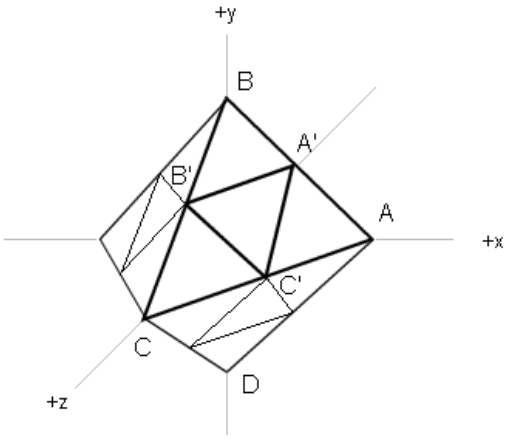
d. Assume that the normals have been specified as in (c). Now switch from OpenGL's Gouraud interpolated shading to Phong interpolated shading. This will affect both the diffuse and specular components of the shading over the surface. How will the diffuse appearance of the image change? How will the specular appearance of the image change?

e. What is one reason that Gouraud shading is standard on most systems and Phong shading is not?

Problem 3 (cont'd)

f. Remember that this object is being used to simulate a sphere. One easy-to-calculate improvement to the geometry of the model is to subdivide each triangular face into four new equilateral triangles and then specify three new additional vertices. If you subdivided triangle ABC this way as shown in the figure below, what would be the best choices for the new coordinates and the normals of the three added vertices A', B', and C' in order to more closely approximate a sphere? Why?

Triangle	Vertex	Coordinates	Normals
A'B'C'	A'		
A'B'C'	B'		
A'B'C'	C'		



g. How would you continue to improve the approximation to the sphere?

Problem 4. 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).

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

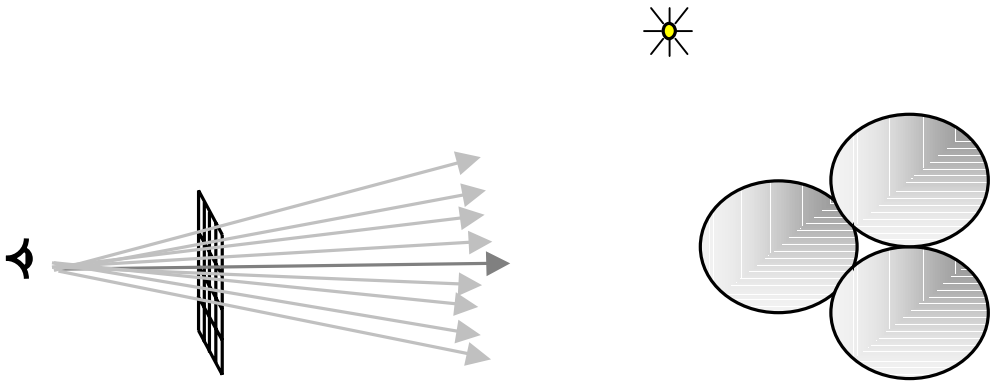
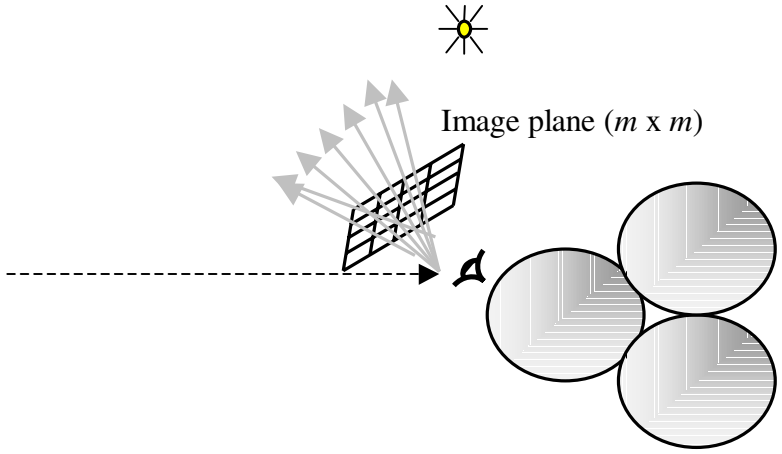


Image plane (n x n)

b. Now let's say we want to capture glossy reflections by spawning many rays at each intersection point roughly in the specular direction. As indicated by the figure below, we can cast these rays by positioning the new viewpoint at a given intersection point, setting up a new image plane of size m x m oriented to align with the specular direction, and then run our modified Z-buffer algorithm again. Let's say we follow this procedure for all pixels for k bounces in a scene assuming non-refractive surfaces. In effect, how many rays will we end up tracing?



Problem 4 (cont'd)

c. If we compute a spread of transmitted rays to simulate translucency as well, how many rays will we end up tracing?

Now, instead of using z buffer algorithm we use the distribution ray tracing method. Count how many rays we need to trace. For each of these cases:

d. First intersections (0 bounces)

e. *k* bounces to simulate gloss only (no refraction/translucency).

f. *k* bounces to simulate both gloss and translucency.

g. Given your answers to (a)-(f) when would it be appropriate to use the modified Z-buffer in a ray tracing algorithm?

Problem 5. Texture filtering

Three approaches to storing and retrieving the information needed for texture mapping that were discussed in class are: direct summation (brute force), mip maps with linear interpolation, and summed area tables. In this problem, assume that we are using these techniques to determine a color value for a pixel p in screen space that is covered by a k by k square of texture elements (texels) in the texture map. The original image that is represented by the texture map is a square of n by n pixels. For each of the techniques, provide estimates of the following values in big O (asymptotic complexity) notation.

a. The number of lookup operations required to retrieve the relevant texels.

Direct summation:

Summed area tables:

Mip maps:

b. The storage required for the texture information.

Direct summation:

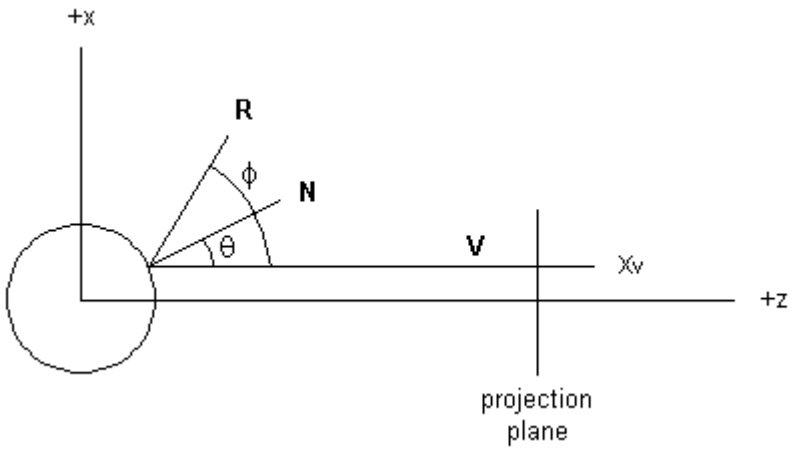
Summed area tables:

Mip maps:

Problem 6. Environment mapping

One method of environment mapping (reflection mapping) involves using a "gazing ball" to capture an image of the surroundings. The idea is to place a chrome sphere in an actual environment, take a photograph of the sphere, and use the resulting image as an environment map. Let's examine this in two dimensions, using a "gazing circle" to capture the environment around a point.

Below is a diagram of the setup. In order to keep the intersection and angle calculations simple, we will assume that each view ray **V** that is cast through the projection plane to the gazing circle is parallel to the z-axis, meaning that the viewer is located at infinity on the z-axis. The circle is of radius 1, centered at the origin.



a. If the x-coordinate of the view ray is x_v, what are the (x,z) coordinates of the point at which the ray intersects the circle? What is the unit normal vector at this point?

b. What is the angle θ between the view ray **V** and the normal **N** as a function of x_v?

Problem 6 (cont'd)

c. Note that the angle ϕ between the view ray **V** and the reflection direction **R** is $2*\theta$. Plot ϕ versus x_v. In what regions do small changes in the intersection point result in large changes in the reflection direction?

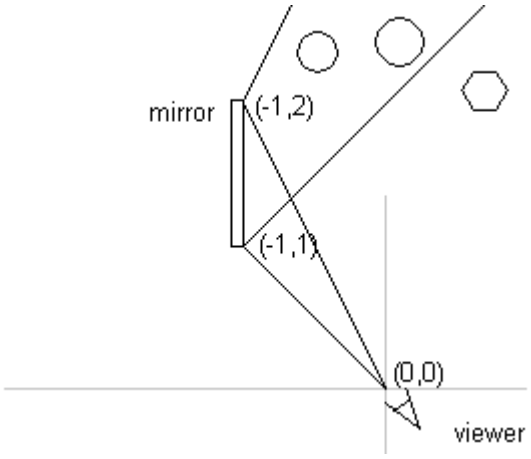
d. If we were ray-tracing and used the reflection ray **R** to index into a photograph of a chrome circle sitting where the reflection circle is now, what types of errors would we expect to see in the resulting image?

Problem 7. More uses of texture mapping.

Texture mapping can be used in many ways in addition to simply providing realistic looking surface material treatments on objects.

For example, the texture applied to a surface doesn't have to be a representation of the surface material at all. It could be an image rendered from another point of view in the model.

a. Consider the scene shown in the figure. The viewer is at the origin, and there is a perfect mirror sitting in the scene nearby. Think about the image in the mirror. Calculate where you would place another center of projection in order to render the image that will be visible in the mirror from the origin, and draw it on the diagram. Show the coordinates of the added center of projection.



b. Write a brief pseudo-code algorithm that describes the steps you would go through to use this idea in a modified z-buffer renderer to provide simple mirrors.