Computer Graphics — Zoran Popovi}
CSE 457 — Autumn 1999

Homework 3

Received: Mon, Nov. 1
Due: Mon, Nov. 15

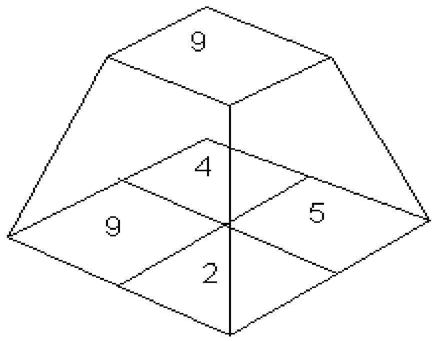
DIRECTIONS

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: ________________________________

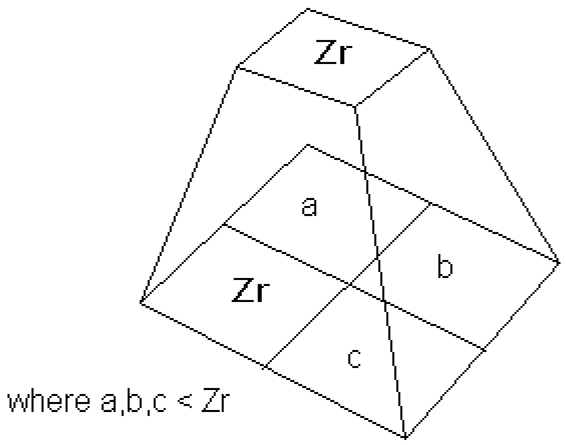
Problem 1. (20 Points)

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) (3 Points) At the coarsest level of the pyramid there is just a single Z value. What does that Z value represent?

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



Problem 1 - continued.

b) (5 Points) What can we conclude if **Zr** < **Zp**?

c) (5 Points) What can we conclude if **Zp** < **Zr**?

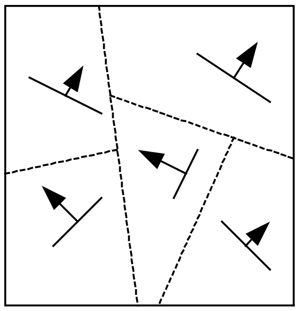
If the visibility test is inconclusive, then the algorithm applies the same test recursively: it goes to the next finer level of the pyramid, where the region **R** is divided into four quadrants, and attempts to prove that polygon **P** is hidden in each of the quadrants **R** of that **P** intersects. Since it is expensive to compute the closest Z value of **P** within each quadrant, the algorithm just uses the same **Zp** (the nearest Z of the *entire* polygon) in making the comparison in every quadrant. If at the bottom of the pyramid the test is still inconclusive, the algorithm resorts to ordinary Z-buffered scan conversion to resolve visibility.

d) (7 Points) Suppose that, instead of using the above algorithm, we decided to go to the expense of computing the closest Z value of **P** within each quadrant. Would it then be possible to always make a definitive conclusion about the visibility **P** of within each pixel, without resorting to scan conversion? Why or why not?

Problem 2. (20 Points)

BSP trees are widely used in computer graphics. There are many variations that can be used to increase performance. The following questions deal with some of these variations.

For the version of BSP trees that we learned about in class, polygons in the scene (or more precisely, their supporting planes) were used to do the scene splitting. However, it is not necessary to use existing polygons – one can choose arbitrary planes to split the scene:



a) (2 Points) What is one advantage of being able to pick the plane used to divide the scene at each step? What is one disadvantage of not just using existing polygons?

Recall that when using a BSP tree as described in class, we must draw *all* the polygons in the tree. This is very inefficient, since many of these polygons will be completely outside of the view frustum. However, it is possible to store information at the internal nodes in a BSP tree that will allow us to easily determine if any of the polygons below that node will be visible. If none of the polygons in that sub-tree will be visible, we can completely ignore that branch of the tree.

b) (5 Points) Explain what extra information should be stored at the internal nodes to allow this, and how it would be used to do this "pruning" of the BSP tree.

Problem 2 - continued.

c) (6 Points) In class, we talked about doing a "back to front" traversal of a BSP tree. But it is sometimes preferable to do a "front to back" traversal of the tree, in which we draw polygons closer to the viewer before we draw the polygons farther away. (See part (d) for one reason why this is useful) How should the tree traversal order be changed in order to do a front to back traversal?

When we traverse a BSP tree in back to front order, we may draw over the same pixel location many times, which is inefficient since we would do a lot of "useless" shading computations. Assume we instead traverse the tree in front to back order. As we scan convert each polygon, we would like to be able to know whether or not each pixel of it will be visible in the final scene (and thus whether we need to compute shading information for that point).

d) (7 Points) What simple information about the screen do we need to maintain in order to know if each pixel in the next polygon we draw will be visible or not?

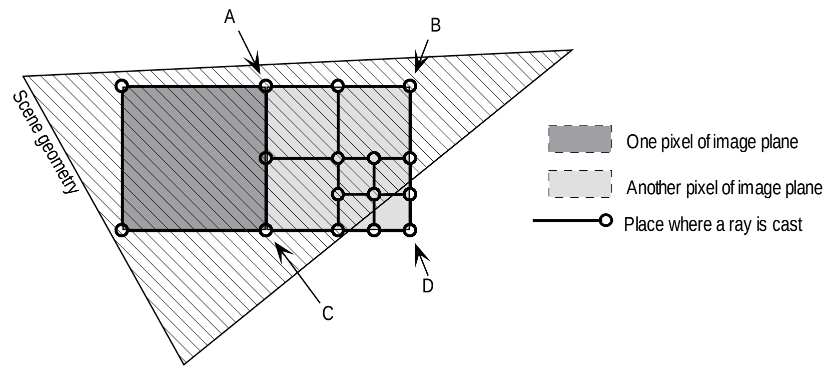
Problem 3. (15 Points)

The company you work for has just bought rights to a raytracing engine. Unfortunately, you don't have the source code, just a compiled library. You have been asked to determine how rays are terminated. So, you call the authors you find out even they don't remember for sure. All they can tell you is this: *The termination criteria for tracing rays is either (a) rays are traced to a maximum recursion depth of 5, or (b) rays are adaptively terminated based on their contribution to a pixel color.*

a) (5 Points) Describe a scene that can be used to determine which method is used. Be specific about all relevant aspects of the scene and what you would look for in the resulting image to determine which termination method is used.

One of the features included in the raytracing engine your company bought is a brand new algorithm for antialiasing by adaptive supersampling.

The normal implementation is to sample rays at the corner of every pixel, compare the colors of each sample, and if the difference between neighboring sample colors is too great, subdivide that region recursively and sample more times. (See the diagram below, or Foley, et al., 15.10.4)



However, in this new algorithm, we subdivide and supersample if neighboring rays *intersect different objects*. In other words, note the light-grey pixel above. Three of the four corner samples (a, b, and c) intersect the scene geometry. The fourth corner (d), misses the geometry completely. So we choose to supersample this pixel without ever comparing colors.

Problem 3 - continued.

b) (10 Points) In what ways is this better than the traditional way? In what ways is it worse?

Problem 4. (25 Points)

The Phong shading model can be summarized by the following equation:

$$I_{phong} = k_e + k_a I_a + \sum_i \left[I_{l_i} \left[k_d (\mathbf{N} \cdot \mathbf{L}_i)_+ + k_s (\mathbf{V} \cdot \mathbf{R}_i)_+^{n_s} \right] \min\left\{ 1, \frac{1}{a_0 + a_1 d_i + a_2 d_i^2} \right\} \right]$$

where the summation *i* is taken over all light sources. The variables used in the Phong shading equation are summarized below:

$I \quad a_0 \quad a_1 \quad a_2 \quad d_i \quad k_e \quad k_a \quad k_d \quad k_s \quad n_s \quad I_a \quad I_{li} \quad \mathbf{L}_i \quad \mathbf{R}_i \quad \mathbf{N} \quad \mathbf{V}$

a) (6 Points) Which of the quantities above are affected if...

- …the viewing direction changes?
- …the position of the *i*th light changes?
- …the orientation of the surface changes? Assume that the change in orientation is about the intersection point.

Blinn and Newell have suggested that, when **V** and **L** are assumed to be constants, the computation of $\mathbf{V} \cdot \mathbf{R}$ 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 **L** and **V**. The specular component is then computed from $(\mathbf{N} \cdot \mathbf{H})_+^{n_s}$ instead of from $(\mathbf{V} \cdot \mathbf{R})_+^{n_s}$.

b) (3 Points) Under what circumstances might L and V be assumed to be constant?

c) (4 Points) How does the new equation using H simplify shading equations?

Problem 4 - continued.

The ambient term in the Phong model is one way to guarantee that all visible surfaces receive some light. Another possibility is to use the "headlamp" method in which a point light source is positioned at the eye, but no ambient term is used.

d) (4 Points) Are these two methods equivalent? If so, explain why. If not, describe a scene in which the results would be clearly different.

e) (4 Points) Describe the relationships between **N**, $\mathbf{L}_i$, and $\mathbf{R}_i$ that would result in a point shaded with the Phong model appearing maximally bright.

f) (4 Points) The equation above is not the only hallmark to Mister Phong's fame. We also talked in class about the difference between two polygon shading methods, one called Phong and one called Gouraud. Describe a scene where the difference between Phong and Gouraud shading would be noticeable.

Problem 5. (12 Points)

(3 Points Each) Respond TRUE or FALSE to each of these statements and ***explain your reasoning***.

______ The Phong model is a physical simulation of the behavior of real-world light.

______ For polished metal, the specular component n_s would be large.

______ A rough surface with many tiny microfacets is likely to have a large diffuse reflection coefficient.

_____ In the Phong model, specular reflection does not depend on viewing angle.

Problem 6. (15 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 x 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

b) (5 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 three methods above, what is the asymptotic time complexity of computing this average (in terms of n and r)?

c) (5 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?