Computer Graphics	Profs. Brian Curless and Steve Seitz
CSE 591	Spring 2001

Quiz #1 Study Guide

Hierarchical Modeling, Projections, and Hidden Surfaces

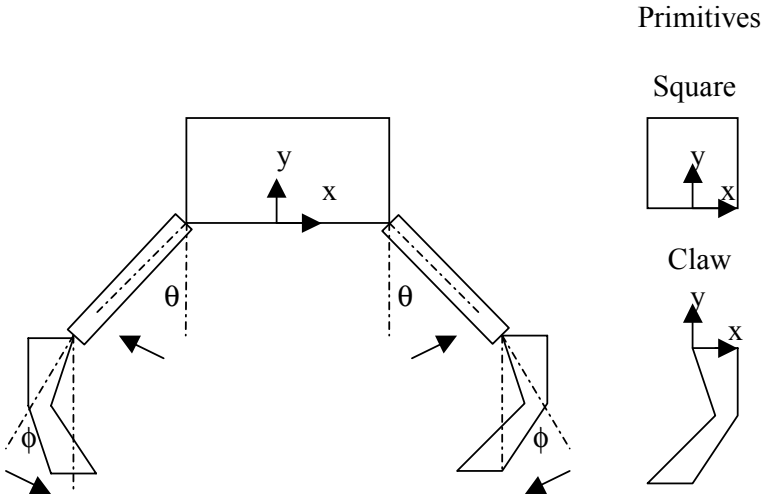
The quiz will be on Monday, May 7. It will cover all material up through hidden surface algorithms. The quiz will last 30 minutes and will be close book. This Study Guide is intended to give you practice thinking in depth about material that was not covered by homework #1.

Problem 1

You are modeling a hanging pincer as shown below. There are two primitives available to you, **Square** and **Claw**. The local coordinates for each primitive are shown. The transformations available are

$R(\theta)$ – rotate by θ degrees (counter-clockwise)
$T(t_x, t_y)$ – translate by t_x, t_y
$S(s_x, s_y)$ – scale by s_x, s_y
R_y – reflect through the y axis

The arrows in the figure indicate the direction of rotation that θ and ϕ refer to for each pincer. The base of the figure is twice as wide (along the x axis) as the square. Each finger is 1/3 as wide as the square, and 3/2 as long.



Primitives

Square

Claw

a. Give names to the different pieces of the pincer. For each piece, define a primitive to draw it if necessary, using, for example, a function definition with OpenGL code. Now draw a directed acyclic graph (DAG) to specify the pincer. For each of the edges in your DAG, write the expression for that transformation using only the symbols given on the previous page. This transformation should be the transformation required to position the piece that is the next node. Each node in the DAG should be one of the primitives given to you, or one of the primitives you have defined. Leave θ and ϕ as symbols, these are the parameters you intend to animate. Assume **Square** is one unit in size.

b. What is the entire sequence of transformations applied to the claw drawn on the far left side of the figure?

Problem 2

a. Perspective Projections

True or False

1. Size varies inversely with distance

2. Distance and angles are preserved

3. Parallel lines do not remain parallel

4. Perspective projections make z-buffers more imprecise

How many different vanishing points can a perspective drawing have?

b. Parallel Projections

True or False

1. Parallel projections are more realistic looking than perspective projections

2. Parallel projections are good for exact measurements

3. Parallel lines do not remain parallel

4. Angles (in general) are not preserved

5. Lengths vary with distance to the eye

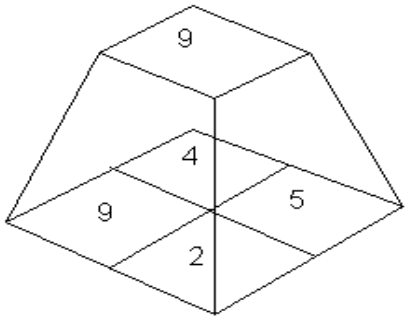
Parallel projections can be further broken down into two types: “orthographic projections”, where the direction of projection is perpendicular to the projection plane, and “oblique projections”, where the direction of projection is not perpendicular to the projection plane.

Two common types of oblique projections are the “cavalier projection” and the “cabinet projection”. In a cavalier projection the foreshortening factors for all three principal directions are equal, whereas in a cabinet projection the edges perpendicular to the plane of projection are foreshortened by one half.

Suppose you wanted to use an oblique projection that foreshortened the edges perpendicular to a plane of projection by one third instead of one half. What angle between the direction of projection and the projection plane is required?

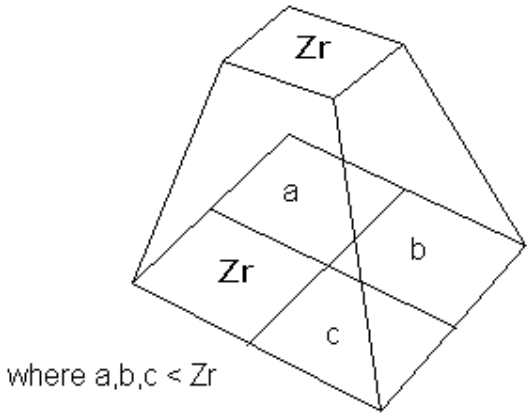
Problem 3

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 3 - 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?

Another type of coherence that one might like to exploit in a hidden surface algorithm is *temporal coherence* - the fact that the same polygons are likely to remain visible from one frame of an animation to the next.

e) Suppose that we kept track of the polygons that were at least partially visible for some frame t. How might we use this information in frame $t + 1$ to exploit temporal coherence with a Z-pyramid? How much better is using a Z-pyramid in this case than using a simple Z-buffer?

Problem 4

Answer the following questions regarding BSP trees

a. Is it sort-first, sort-last, or both?

b. Is it image-precision, object-precision, or both?

c. Is it image-order, object-order, or both?

d. Do all polygons need to be rendered?

e. Is it an online algorithm?

f. Can it handle transparency?

g. Can it handle refraction?

h. Can shading be done efficiently?

i. What are the memory requirements proportional to?

j. What types of 3d graphics programs are well suited to BSP trees?