Written Assignment W2

Computer Graphics 2 (15-463)

Due 5:00pm Thursday, 20 Mar. 1997, at my office (DH 4301A)

1. [20 pts]

This problem analyzes uniform grid subdivisions for ray casting. Assume that there are s surfaces uniformly distributed in a box which is divided into an $n \times n \times n$ grid of rectangular voxels, and the voxels are bigger than the surfaces for the values of n of interest, so that each surface is listed in no more than 8 voxels' lists. Let the time to step from one voxel to its neighbor be t_s, and the time to intersect a ray with a surface be t_i. What is the worst case time cost of intersecting a ray with the surfaces to find the first intersection point, as a function of n, s, t_s, and t_i? Graph cost as a function of n, qualitatively. State your assumptions clearly.

2. [20 pts.]

(It's OK to use a symbolic algebra system for this, but if you do, include a transcript, and explain what you're doing step by step.) You are hardware designer for the new Nintendo *Z Boy* game machine. You need to figure out how many bits of z to use in the z-buffer. Fewer bits mean a cheaper machine. The game programmers want to set the near and far clipping planes at depths of 1mm and 10 meters, in world space, and they want 1mm z-buffer depth resolution, or better, throughout. This means that no two points whose world space depths differ by more than this amount should round to the same integer screen space z value. You suspect the programmers don't understand z-buffers very well, the way they're planning to set the near clipping plane so close.

The transformations from world space to screen space that you'll use are given by the Scene commands[1]:

```
screensize a b d
zrange n f
persp 90 a/b
```

[1]The commands "`zrange` n f" followed by `persp`, map $z = n$ to $z = -1$ (near clipping plane) and $z = f$ to $z = 1$ (far clipping plane). Typically, objects before the near or beyond the far clipping planes are clipped away.

By referring to the `scene.c` and `matrix.c` source files[2], write out the formula to transform a world space point (x, y, z) to screen space, yielding (x_s, y_s, z_s). (Rather than use (x_w, y_w, z_w) to denote world space, I suggest this simpler notation for this problem.) Your formula for screen space coordinates will have the form:

$$\begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} = \text{homodiv}\Big(\mathbf{M} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}\Big)$$

where $\mathbf{M}$ is a product of three 4×4 matrices corresponding to the three transformation commands above, and "homodiv" does homogeneous division:

$$\text{homodiv}\Big(\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}\Big) = \begin{pmatrix} x/w \\ y/w \\ z/w \end{pmatrix}$$

a) Give a formula for $\mathbf{M}$, where each of the 16 elements of the matrix will be a **simplified** expression in the variables d, n, f, a, and b. The final formula should not involve matrix algebra.

b) How does z transform? Find the formula for $z_s(z)$, that is, screen space depth as a function of world space depth. Check that $z_s(n) = -d/2$ and $z_s(f) = d/2$.

c) Define the *z-buffer resolution* to be the change in world space depth corresponding to a unit change in screen space depth. Since z_s is a nonlinear function of z, the z-buffer resolution is not constant, but a function of depth. What is the z-buffer resolution at the near and far clipping planes? That is, what is the slope of the inverse function $z(z_s)$ at $z_s = -d/2$ and $z_s = d/2$?

d) Is 16 bits enough to meet the programmers' request? For 16 bits, $d = 65536$, and z_s values lie in the range [-32768,32768). When $n = .001$ and $f = 10$, what is the z-buffer resolution at the far clipping plane?

e) Would moving the near clipping plane solve the problem? What's the smallest value of n that would give you 1mm z-buffer resolution, without changing f or d?

f) Plot $z_s(z)$ for the values of n in **d)** and **e)** .

Tip: `gnuplot` is an easy-to-use plotting program. Type the following to get a demo of what it can do:

```
<setenv DISPLAY appropriately>
```

[2] Or you could use the C++ versions, but these are a bit harder to follow because of the large matrix library.

```
gnuplot
set xlabel "x axis"
set ylabel "y axis"
f1(z) = sin(z)
f2(z) = cos(z)+cos(3*z)
plot [z=-pi:pi] [-2.5:2.5] f1(z) title "f1", f2(z) title "f2"
```

For documentation on gnuplot, type "`help`" from within gnuplot, or see `http://arachnid.cs.cf.ac.uk/Latex/Gnuplot/index.html`. For hard-copy plots, try `gnuplot`'s "`set terminal`" command.

g) With $n = .001$, how many bits in your z-buffer would be needed to get 1mm z-buffer resolution?

modified 18 Mar 97 to clarify question 2f.