Written Assignment W2

Computer Graphics 2 (15-463)
14 Mar. 1996

Due 11:59pm Thursday, 21 Mar. 1996, at my office (DH 4301A)

1. [10 pts.]

Prove that perspective transformation preserves lines, that is, that lines transform to lines, not curves, through a perspective transformation. This is an important property that allows screen space z to be linearly, and quickly, interpolated in z-buffer algorithms.

One way to prove this is as follows (if you find another way, we'll accept that too!). Define a line in some "object space" through the object space points $\mathbf{p}^o = (p_x^o, p_y^o, p_z^o, 1)^T$ and $\mathbf{q}^o = (q_x^o, q_y^o, q_z^o, 1)^T$. Superscript "o" denotes object space, not exponentiation. A point in object space on the line through them is $\mathbf{x}^o(\alpha) = \mathbf{p}^o + \alpha(\mathbf{q}^o - \mathbf{p}^o)$, where α is a free parameter. Transform this line into "screen space" using an arbitrary 4×4 matrix $\mathbf{M}$, yielding some curve.

Meanwhile, transform just the points $\mathbf{p}^o$ and $\mathbf{q}^o$ to homogeneous screen space like so: $\mathbf{p} = (p_x, p_y, p_z, p_w)^T = \mathbf{M}\mathbf{p}^o$, and $\mathbf{q} = (q_x, q_y, q_z, q_w)^T = \mathbf{M}\mathbf{q}^o$. Note that these are homogeneous coordinates before division by w. To find the line through them, we first do homogeneous division on each, then interpolate: $\mathbf{y}(\beta) = \mathbf{p}/p_w + \beta(\mathbf{q}/q_w - \mathbf{p}/p_w)$, where β is a free parameter. (I'm leaving superscript "s" off these screen space points for simplicity).

We want to show that the "curve" and the latter line are everywhere coincident. Points on the "curve", in screen space, are found by transforming the object space point $\mathbf{x}^o(\alpha)$ by $\mathbf{M}$ and doing the homogeneous division:

$$\frac{\mathbf{M}\mathbf{x}^o(\alpha)}{[\mathbf{M}\mathbf{x}^o(\alpha)]_w} = \frac{\mathbf{p} + \alpha(\mathbf{q} - \mathbf{p})}{\mathbf{p}_w + \alpha(\mathbf{q}_w - \mathbf{p}_w)}$$

for any α. Points on the line, in screen space, are given by $\mathbf{y}(\beta)$, above. Equate a point on the curve with a point on the line, and solve for the function $\beta(\alpha)$. Note that the parameterization is nonuniform; $\beta(\alpha)$ is nonlinear. Write out your derivation neatly, explaining your steps. If there's always such a β, then the curve is a line.

2. [20 pts.]

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 1cm 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]:

$$
\begin{array}{l}
\texttt{screensize}\ \ a\ \ b\ \ d \\
\texttt{zrange}\ \ n\ \ f \\
\texttt{persp 90}\ \ a/b
\end{array}
$$

By referring to the `scene.c` source file, 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) How does z transform? Find the formula for $z_s(z)$, that is, screen space depth as a function of world space depth. It will have parameters d, n, and f. Simplify it as much as possible. The final formula should not involve matrix algebra. Check that $z_s(n) = -d/2$ and $z_s(f) = d/2$.

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

b) 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 $z_s = d/2$?

c) 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?

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

e) Plot the curves for **c)** and **d)** .

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

```
<setenv DISPLAY appropriately>
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"
```

I couldn't find full documentation on `gnuplot` on Andrew, but there is some at `http://arachnid.cs.cf.ac.uk/Latex/Gnuplot/index.html` . For hard-copy plots, try `gnuplot`'s "`terminal`" command.

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