Lecture 9: Supplemental Material

Assignment 4 asks you to fill in several missing elements to a polygon scan converter. For each polygon edge, you are given two vertices. Each vertex is described by a data structure including:

the vertex x and y, in screen space;

the vertex z, in eye space;

the vertex "color," derived by the application of a local illumination model; and

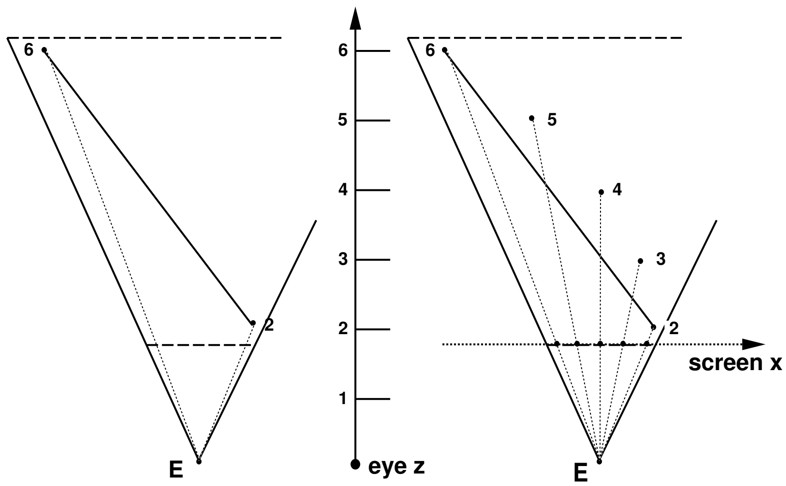
other data necessary only for extra credit tasks.

Recall that Gouraud interpolation calls for the interpolation of color values across polygon (span) interiors. Gouraud interpolation of colors simply performs interpolation in screen space. (This method is invariant under rotation in screen space, but not under general rotations; thus it is arguably incorrect. See Lecture 9, page 9 to see why. Nevertheless, you will find it in use in just about every graphics pipeline in existence.)

Now let's consider the interpolation of depth values. Note again what you are given for each vertex: its x and y coordinates in screen space, but its z coordinate in *eye space*. (I'll use suffixes S and E, respectively for these spaces in what follows.) Again, it is incorrect to interpolate eye space quantities in screen space. That is, if one has moved α of the way across a span from pixel x_{S1} to pixel x_{S2}, to the pixel $x_S = x_{S1} + \alpha x_{S2} - x_{S1}$, one might naively compute the quantity

$$z_{E1} + \alpha(z_{E2} - z_{E1})$$

and try to use it to compare depths from different spans. But **this will not work**. Study the diagram below (from Lecture 9, page 23) to understand why.



Instead, we have to do something more sophisticated. We can work out what must be done by using the Perspective Transformation Matrix from Lecture 6, page 21:

$$\mathbf{M} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

To make our lives easier, let's choose some simple values for the frustum bounding planes. (All of the following derivations go through even if we leave things generic, but it would be hard to work them by hand.) I'll choose the left and bottom planes at $l = b = -1$; the right and top planes at $r = t = +1$, and the near and far planes at $n = 1$ and $f = 2$. Then the perspective matrix becomes

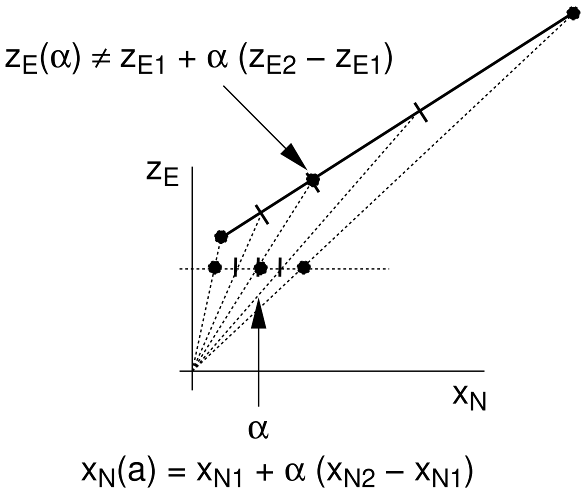
$$\mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -3 & -4 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

This matrix takes points expressed in eye space (suffix E) and maps them to normalized device coordinates (suffix N). Look at how $\mathbf{M}$ is used to compute x_N, given an eye space point $x_E, y_E, z_E, 1$:

$$x_N = \frac{x_E}{-z_E} \tag{1}$$

(Note how the $-z_E$ shows up in the denominator, due to the homogeneous division described in Lecture 6.)

Now, in the assignment code, what are you *given*? At two endpoints of each edge, NDC x and y (actually, screen space x and y, but these are just NDC coordinates scaled and offset), and eye space z. That is, you are given x_{N1}, x_{N2}, z_{E1}, and z_{E2}. During scan conversion, when you are α of the way from x_{N1} to x_{N2}, you must compute $z_E(\alpha)$; that is, you must find the eye space z value corresponding to $x_N(\alpha)$.



It is possible to express $z_E(\alpha)$ solely in terms of z_{E1}, z_{E2}, and α. You might wish to tackle this yourself. If not, the following should serve to guide you through the derivation.

First, use Eqn. 1 to express $z_E(\alpha)$ as:

$$z_E(\alpha) = \frac{x_E(\alpha)}{-x_N(\alpha)} \tag{2}$$

We also know two boundary conditions from Eqn. 1, namely that

$$\begin{aligned} x_{N1} &= \frac{x_{E1}}{-z_{E1}} \text{ and} &(3)\\ x_{N2} &= \frac{x_{E2}}{-z_{E2}} &(4)\end{aligned}$$

Also, we know that z_E and x_E are linearly dependent (why?), so

$$\frac{z_E(\alpha) - z_{E1}}{z_{E2} - z_{E1}} = \frac{x_E(\alpha) - x_{E1}}{x_{E2} - x_{E1}} \tag{5}$$

Finally, we are interpolating between x_{N1} and x_{N2} using the parameter α:

$$x_N(\alpha) = x_{N1} + \alpha(x_{N2} - x_{N1}) \tag{6}$$

Now, substitute Eqn. 6 into the denominator of Eqn. 2. In Equation 5, solve for $x_E(\alpha)$, plug the resulting expression into the numerator of Eqn. 2, and rearrange to isolate $z_E(\alpha)$.

You should now have an expression for $z_E(\alpha)$, solely in terms of $z_{E1}, z_{E2}, x_{E1}, x_{E2}, \alpha, x_{N1}$, and x_{N2}. Eliminate the NDC terms by substitution from Eqns. 3 and 4. You should find that x_{E1} and x_{E2} magically disappear, leaving you with an expression for z_E solely in terms of z_{E1}, z_{E2}, and α.

You have now done most of what we set out to do, by expressing z_E this way. However, note that your expression is not suitable for *incremental* evaluation.

To fix this (and to make all this algebra pay off in your scan conversion program), there remains just a little more algebra. Define two auxiliary variables h_1 and h_2 as

$$\begin{aligned} h_1 &= \frac{1}{z_{E1}} \text{ and}\\ h_2 &= \frac{1}{z_{E2}} \end{aligned}$$

Now express z_E solely in terms of h_1, h_2, and α. This final expression is the one you will need for assignment 4.

To use it, compute h_1 and h_2 at setup time (when preparing an edge or span for interpolation), then use your expression to compute z_E from h_1, h_2, and α at each pixel. Most likely you will compute $\Delta_h = \frac{h_2 - h_1}{x_{N2} - x_{N1}}$, initialize a running h value (to what?), then add Δ_h to it each time you advance by one pixel.