Pseudodepth in the Z-buffer

CSC 418 Tutorial
Friday February 13, 2004

Recall the Z-buffer algorithm:

```
for each polygon
        for each pixel indexed by i, j
                P ← point on polygon at this pixel
                if (depth(P) < depth[i][j])
                        depth[i][j] ← depth(P)
                        color[i][j[ ← color(P)
                end if
        end if
end for
```

Assume right-handed eye space.
The point P projected onto the image plane (i.e. near plane) is

$Q = \{-(fP_x)/P_z, -(fP_y)/P_z\}$

The depth is

$\text{depth}(P) = \text{sqrt}(P_x^2, P_y^2, P_z^2)$

This is an expensive calculation for every fragment (what is a fragment?), so we replace it with an easier calculation, using "psedudodepth" (δ).

What about just using P_z, the distance from the plane perpendicular to the view vector through the eye?

If we keep a consistent denominator among Qx, Qy, and Qδ (pseudodepth), we can take advantage of homogeneous coordinates to encode perspective projection as a 4x4 matrix.

The goals for pseudodepth:

- The denominator should be $- P_z$
- The numerator should be easy to compute, i.e. a linear function of P_z
- The pseudodepth should be -1 for points on the near plane. Why?
- The pseudodepth should be 1 for points on the far plane. Why?

So $\text{pseudodepth}(-P_z) = (a\,P_z + b)/(-P_z)$ for some a, b
Given goals, 3, 4, solve for a, b:

$a = -(F + f) / (F - f)$

$b = -2Ff / (F - f)$

Note the following:

- Pseudodepth is a nonlinear function of P_z.
- Points closer to the near plane have the highest pseudodepth resolution.

Points closer to the far plane have the lowest pseudodepth resolution.

Resulting perspective transformation matrix:

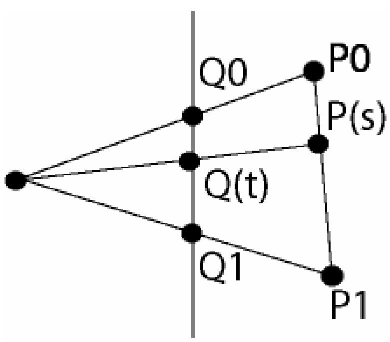
$$Q = \begin{vmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{vmatrix} * P = \begin{vmatrix} fPx \\ fPy \\ aPz + b \\ -Pz \end{vmatrix}$$

This still needs to take into account remapping Px and Py to the normalized view volume, but this is sufficient for the remaining discussion.
What is Q after perspective divide? Denote it Q = [Qx, Qy, Qδ, 1]

Using pseudodepth during scan conversion:

When scan converting a polygon, the linear nature of the pseudodepth numerator can be used to avoid calculating the perspective transformation of each fragment.

As a single example, consider the scan conversion code for a horizontal line (P0, P1) in eye space and its perspective transformation (Q0, Q1):



The eye space line is parameterized by s, and the perspective transformation is parameterized by t.

Here is scan conversion code that includes an incremental update of the pseudodepth δ:

```
δ ← Q0δ
deltaδ ← (Q1z – Q0z) / (Q1x – Q0x)
for x ← Q0x to Q1x
    δ += deltaδ
    if (δ < depth(x, y))
        update depth and color buffers
    end if
end for
```

This speed-up extends straightforwardly for nonhorizontal lines, and also to polygons as with normal scan conversion.

Why does this work?

Assume $a = 0$, $b = 1$, $f = 1$
Consider t in [0, 1], deltaT = 1/(Q1x – Q0x)
and s in [0, 1], along eye space line
and corresponding incremental steps along δ, as above.

We need to show $Q\delta(t)$ is a linear function of t. What is $Q\delta(t)$?

$$Q\delta(t) = -1/Pz(s) = 1 / (P0z*(1-s) + P1z*s) = 1 / (P0z + s * (P1z - P0z))$$

What is s? For every point P(s), there is a corresponding projection Q(t). In other words, for each s there is a corresponding t. However, the correspondence is complicated—this derivation finds that correspondence.

Consider the mapping of Q(t) to P(s):
$Qx(t) = - Px(s) / Pz(s)$; solve for s:

$$Qx(t)*P0z - Qx(t)*P0z*s + Qx(t)*P1z*s = -P0x + P0x*s - P1x*s$$

$$Qx(t)*P0z + P0x = s* [Qx(t) * P0z - Qx(t) * P1z + P0x - P1x]$$

$$s = (Qx(t)*P0z + P0x) / (Qx(t)*(P0z - P1z) + P0x - P1x)$$

Going back to the previous expression for $Q\delta(t)$:

$$Q\delta(t) = 1 / \{P0z + [Qx(t)*P0z + P0x) / (Qx(t)*(P0z - P1z) + P0x - P1x)] * (P1z - P0z)\}$$

$$= [Qx(t)*(P0z - P1z) + P0x - P1x] / [P0z*(Qx(t)(P0z - P1z) + P0x - P1x) + (Qx(t)*P0z + P0x)*(P1z - P0z)]$$

$$= Qx(t) * (P0z - P1z)/(P0x*(P1z - P0z)) + (P0x-P1x) / (P0x*(P1z-P0z))$$

Qx(t) is the only term that changes with respect to t, and it is also linear with respect to t, so $Q\delta(t)$ is linear, hence the pseudodepth is linear in image space.

Why not increment Pz, the eye space depth?

Pz is not a linear function of image space. From above, the image space point {Qx, Qy} is {-fPx/Pz, -fPy/Pz}, a nonlinear relation ship between both Qx and Qy (the scan-converting domain) and Pz.

Why isn't Z linear in screen-space? Considered a foreshortened view of a building (or a checkerboard), which has windows along the side. The windows close by occupy many pixels, the ones far away are tiny. Stepping along pixels will correspond to small steps in Z for the nearby windows, and large steps in Z for the faraway windows.

For a triangle a great deal of computation can be saved:

Incrementally update pDepth along two sides as scan-conversion progresses along scan lines. Scan convert each scan line between these two sides: