Choosing a Projection Matrix
ECS 175 5/21/2011

One projection matrix that works well with z-buffering is:

\[
\begin{bmatrix}
1 & 0.0 & 0.0 & 0.0 \\
0.0 & 1.0 & 0.0 & 0.0 \\
0.0 & 0.0 & -2/n & -3 \\
0.0 & 0.0 & -1/n & 0.0
\end{bmatrix}
\]

This takes the frustum between \( z = -n \) and \( z = -3n \) to the Normalized Device Coordinates cube which extends between \((-1, -1, -1)\) and \((1, 1, 1)\). Notice that the point \((x, y, -n, 1)\) goes to \((x, y, -1, 1)\) and the point \((x, y, -3n, 1)\) goes to \((x/3, y/3, 1, 1)\).

Generalizing this a little, we can consider the frustum between \(-n\) and \(-kn\) for any \(k\). The matrix should be:

\[
\begin{bmatrix}
1 & 0.0 & 0.0 & 0.0 \\
0.0 & 1.0 & 0.0 & 0.0 \\
0.0 & 0.0 & \frac{k+1}{(k-1)n} & \frac{-2k}{k-1} \\
0.0 & 0.0 & -1/n & 0.0
\end{bmatrix}
\]

Notice where it takes the points \((x, y, -n, 1)\) and \((x, y, -kn, 1)\).

In the most general case, the screen is not the square between \((-1, -1)\) and \((1, 1)\) but an arbitrary rectangle between \((l, b)\) and \((r, t)\), lying in the plane \(z = -n\) and with the frustum extending to \(z = -f\) ("f" for far). Taken from the documentation for the “old” OpenGL glFrustum() function, here is a formula for a completely general projection matrix:

\[
\begin{bmatrix}
\frac{2n}{r-l} & 0.0 & \frac{r+l}{r-l} & 0.0 \\
0.0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0.0 \\
0.0 & 0.0 & \frac{-f+n}{f-n} & -\frac{2fn}{f-n} \\
0.0 & 0.0 & -1.0 & 0.0
\end{bmatrix}
\]

The simpler matrices above are the special cases, divided through by \(n\) (notice we can divide through by whatever we like, since the result will be normalized by the projective divide afterwards).

How to choose \(n\)? A frustum with \(n = 3\), for example, with exhibit a fair amount of perspective, like a object near your face. A frustum with \(n = 7\) or so feels much more natural for a virtual world.