Homogeneous Coordinates

We usually think of a 3D point as being represented by a triple: \((x,y,z)\). Using homogeneous coordinates, we add a 4th number: \((x,y,z,w)\). A graphics system, by convention, performs transformations and clipping using \((x,y,z,w)\) and then divides \(x, y,\) and \(z\) by \(w\) before it uses them.

Thus \((1,2,3,1)\), \((2,4,6,2)\), and \((-1,-2,-3,-1)\) all represent the same 3D point.

Another Reason is to be able to Represent Points at Infinity

This is useful to be able specify a parallel light source by placing the light source location at infinity. The point \((1,2,3,1)\) represents the 3D point \((1,2,3)\). The point \((1,2,3,\frac{1}{2})\) represents the 3D point \((2,4,6)\). The point \((1,2,3,\frac{1}{100})\) represents the point \((100,200,300)\). Points at infinity are used for parallel light sources and some shadow algorithms.

However, When Using Homogeneous Coordinates, You Sometimes Just Need to be able to get a Vector Between Two Points

To get a vector between two homogeneous points, we subtract them:

\[
\begin{align*}
(x_2, y_2, z_2, w_2) - (x_1, y_1, z_1, w_1) &= \left(\frac{x_2 - x_1}{w_2}, \frac{y_2 - y_1}{w_2}, \frac{z_2 - z_1}{w_2}\right) \\
&= \left(x_2 - x_1, y_2 - y_1, z_2 - z_1\right)/w_2 \\
\end{align*}
\]

Fortunately, most of the time that we do this, we only want a unit vector in that direction, not the full vector. So, we can ignore the denominator, and just say:

\[
\vec{v} = \text{normalize}(w_2 x_2 - w_1 x_1, w_2 y_2 - w_1 y_1, w_2 z_2 - w_1 z_1);
\]

Here’s the code to do this:

```
float VectorBetween(vec3 v1, vec3 v2)
{
    return normalize(v2 - v1);
}
```