Reverse Perspective Transformation

Jed Margolin

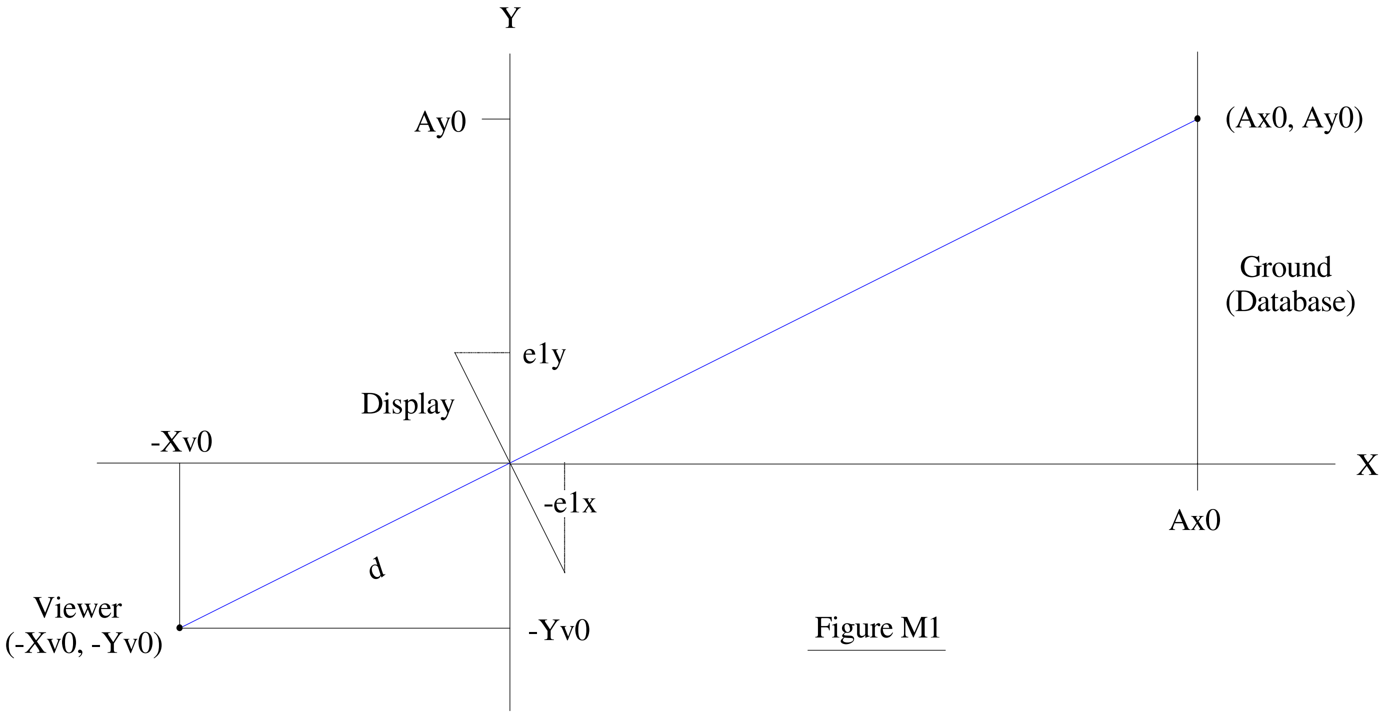

Figure M1

As shown in Figure M1 the Viewer is at coordinates (-Xv0, -Yv0).

The Ground is at the line x = Ax0. The Ground is actually the Database Address. For this 2D example, it is only a line.

The display is centered at the Origin and has the slope -e1y/e1x. The display is perpendicular to the Viewer's Boresight (line of sight vector). The distance between the Viewer and the center of the Monitor is 'd'.

The Viewer's Boresight passes through the Origin (the center of the display) and intersects the Ground at coodinates (Ax0,Ay0).

The Viewer's coordinates are always referenced to the coordinate system. There is no separate Viewer coordinate system. When the Viewer moves, the Display moves with the Viewer and everything (Display and Ground) is translated to place the center of the display back to the Origin. This is also true if the Viewer were to change the Boresight.

For each Viewer position and Boresight, the following are constants:

1. The Viewer's position (-Xv0,-Yv0).

2. The Viewer's Boresight.

3. The location of the Ground (x= Ax0).

We want to know, for a given displacement (position) on the display, where would the line from the Viewer that passes through that position continue on to intersect the Ground?

Referring to Figure M2:

The equation for a line is: y = m*x + b.

The slope of the Boresight line is: Yv0/Xv0. The Boresight line intersects the Origin so b = 0.

Therefore, for the Boresight Line: **y = Yv0/Xv0 * x** .

The slope of the line that defines the monitor is: -e1y/e1x. It also passes through the Origin.

Therefore, for the Display line: **y = -e1y/e1x * x** .

Figure M2 is a better look at the display.

For displacement **dsy** along the vertical face of the Monitor we want to determine (**sx,sy**).

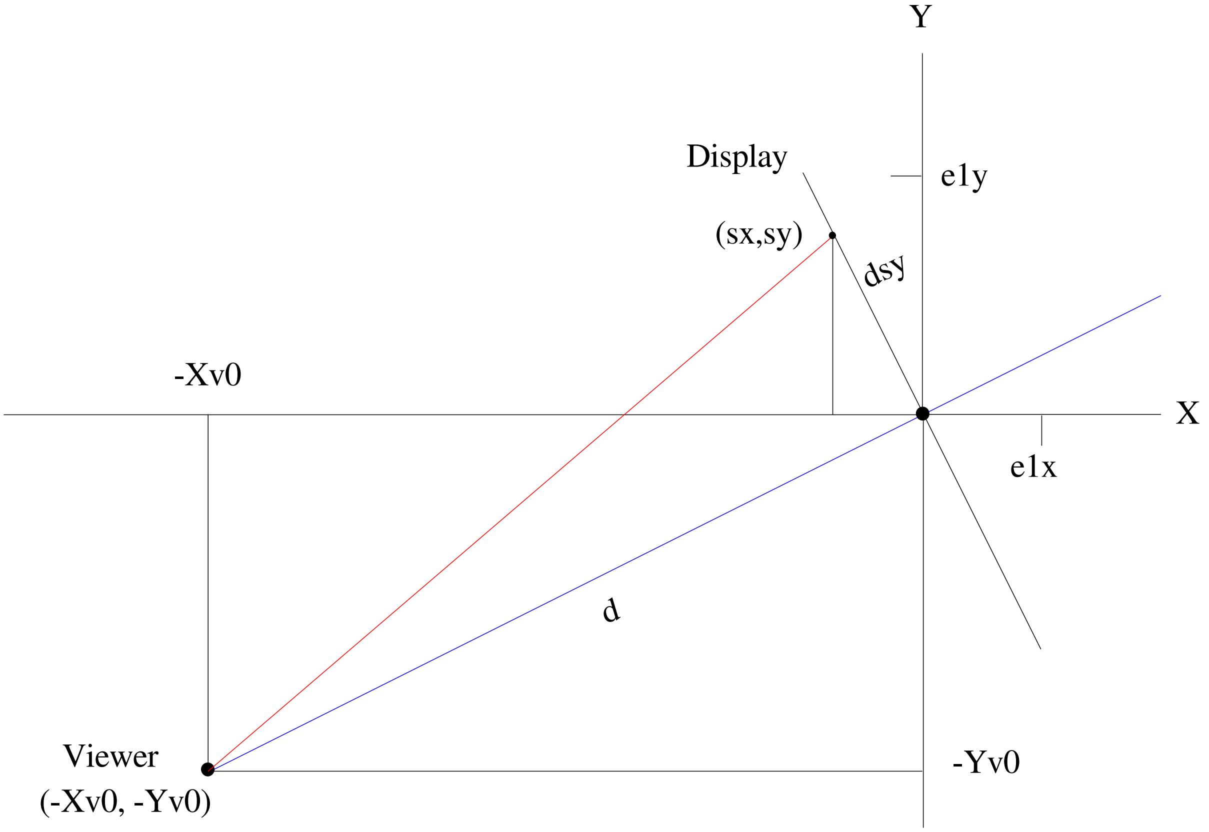

Figure M2

Let’s zoom in more on the Display in Figure M2 by looking at Figure M3.

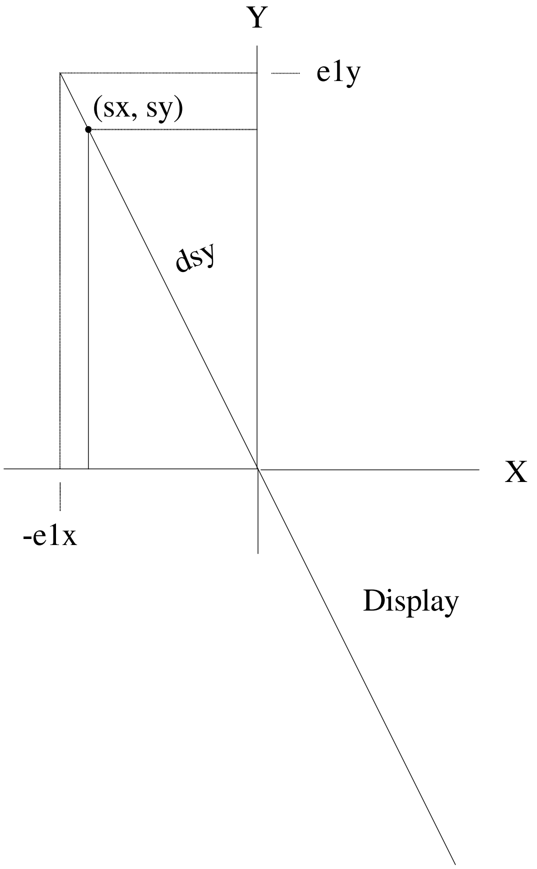

Figure M3

We know that $\mathbf{dsy^2 = sx^2 + sy^2}$

The equation for the Display Line is y = m*x + b where **m = e1y/e1x** and **b = 0**.
For now we will use **y = m*x** .

Slope **m** is also the slope of **sy/sx**.

$$dsy^2 = sx^2 + sy^2$$

so

$$dsy^2 = sx^2 + (m*sx)^2 \qquad // [sy = m*sx]$$

so that

$$dsy^2 = sx^2 + m^2 * sx^2$$

$$= sx^2 (1 + m^2)$$

$$sx^2 = dsy^2/(1+m^2)$$

$$\mathbf{sx = dsy / \sqrt{(1+m^2)}}$$

and therefore, since

sy = m *sx **// m = e1y/e1x**

Substituting for **m** gives:

sx = dsy / √[1+(e1y/e1x)^2]

sy = [e1y/e1x] * sx = [e1y/e1x] * dsy / √[1+(e1y/e1x)^2]

We will note that **dsy / √[1+(e1y/e1x)^2]** is common to both terms (it's **sx**) so, to make the following work easier we will call **w = dsy / √[1+(e1y/e1x)^2]**

sx = w

sy = [e1y/e1x] * w

Knowing the Display vector (**e1x, e1y**) and the displacement **dsy** along the vertical face of the Monitor we have calculated the point (**sx, sy**) on the Display.

With that we can calculate the slope of the line from the Viewer to that point.

The slope of the line is:

$$m2 = \frac{(sy - Yv0)}{(sx - Xv0)}$$

(We are calling it **m2** to distinguish it from the previous **m**.)

$$m2 = \frac{[(e1y/e1x) * w - Yv0]}{(w - Xv0)}$$

This makes m2 a function with dsy as the only independent variable.

We have a point on the line, namely (Xv0,Yvo), so we can solve:

$$Yvo = m2 * Xv0 + b$$

so

$$b = Yv0 - (m2 * Xv0)$$

$$b = Yv0 - \frac{(sy - Yv0)}{(sx - Xv0)} * Xv0$$

since $sy = (e1y/e1x * w)$ and $sx = w$

$$b = Yv0 - \frac{[(e1y/e1x * w) - Yv0]}{(w - Xv0)} * Xv0$$

We will repeat it to give it the same color,

$$b = Yv0 - \frac{[(e1y/e1x * w) - Yv0]}{(w - Xv0)} * Xv0$$

and repeat m2 for the same reason.

$$m2 = \frac{[(e1y/e1x) * w - Yv0]}{(w - Xv0)}$$

Therefore, the point where the line intersects with the Ground (at Ax0) is

$$Ay = m2 * Ax0 + b$$

$$= \frac{[(e1y/e1x) * w - Yv0] * Ax0}{(w - Xv0)} + \frac{Yv0 - \{[e1y/e1x * w - Yv0]\} * Xv0}{(w - Xv0)}$$

We can put them all over the same denominator,

$$= \frac{[(e1y/e1x) * w - Yv0] * Ax0 + Yv0 - \{[e1y/e1x * w - Yv0]\} * Xv0}{(w - Xv0)}$$

And notice there are some common terms.

$$\frac{[(e1y/e1x) * w - Yv0] * Ax0 + Yv0 - \{[e1y/e1x * w - Yv0]\} * Xv0}{(w - Xv0)}$$

Rearrange them slightly.

$$\frac{[(e1y/e1x) * w - Yv0] * Ax0 - \{[e1y/e1x * w - Yv0]\} * Xv0 + Yv0}{(w - Xv0)}$$

$$Ay = \frac{[(e1y/e1x) * w - Yv0] * [Ax0 - Xv0] + Yv0}{(w - Xv0)}$$

Where $w = dsy / \sqrt{[1+(e1y/e1x)^2]}$

If the Display is perpendicular to the Boresight, the slope e1y/e1x is found by rotating the Boresight vector by 90 degrees and normalizing it as shown in Figure M4.

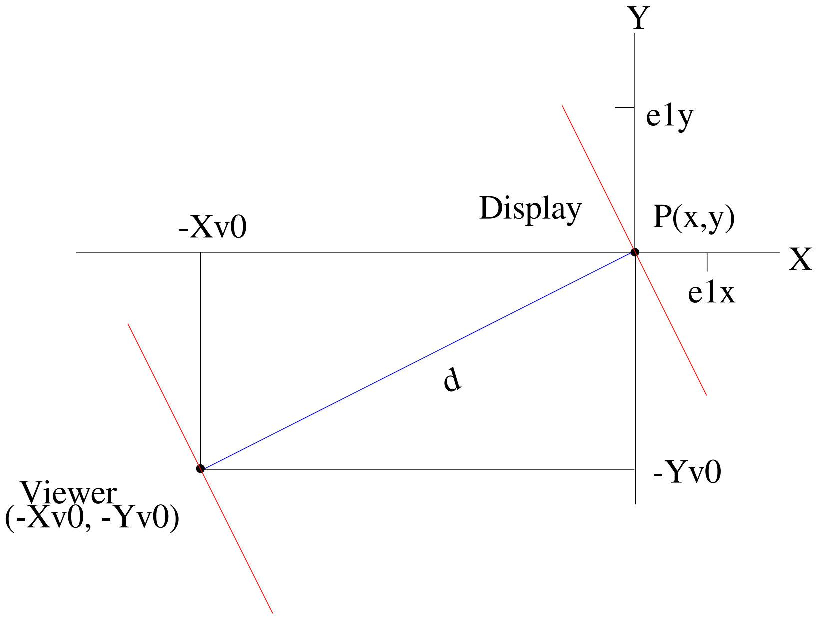

Figure M4

Translate the Viewer to the Origin. The point that had been at the origin, P(0,0) becomes p(Xv0, Yv0). Rotating it by 90 degrees.

$$X' = X*COS(a) - Y*SIN(a)$$

$$Y' = X*SIN(a) + Y*COS(a)$$

Sin(90) = 1, Cos(90) = 0

$X' = X*0 - Y*1$

$Y' = X*1 + Y*0$

So the point P becomes P(-Yv0, Xv0) with px = -Yv0 and py = Xv0.

Normalizing it to produce Normalized Px and Normalized Py (Npx and Npy):

$NPx = -Yv0/\sqrt{[Xv0^2 + Yv0^2]}$

$NPy = Xv0/\sqrt{[Xv0^2 + Yv0^2]}$

nPx is e1x0.

nPy is e1y0.

For a Display of height **h**, they become **h*nPx** and **h*nPy**.

$h*nPx = -Yv0/\sqrt{[Xv0^2 + Yv0^2]} * h$

$h*nPy = Xv0/\sqrt{[Xv0^2 + Yv0^2]} * h$

$$e1y0/e1x0 = \frac{h * -Yv0/\sqrt{[Xv0^2 + Yv0^2]}}{h * Xv0/\sqrt{[Xv0^2 + Yv0^2]}} = \frac{-Yv0}{Xv0} = -Yv0/Xv0$$

$$w = dsy / \sqrt{[1+(e1y/e1x)^2]} = dsy / \sqrt{[1 + (Yv0/Xv0)^2]}$$

$$Ay = \frac{[-Yv0/Xv0 * w - Yv0] * [Ax0 - Xv0] + Yv0}{(w - Xv0)} \qquad \text{[Equation M1]}$$

Where $w = dsy / \sqrt{[1+ (Yv0/Xv0)^2]}$

$\sqrt{[Xv0^2 + Yv0^2]}$ is the distance from the Viewer to the center of the display (which Möller calls ‘d’) and is built into the Xv0 and Yv0 coordinates. If Xv0 and Yvo change. the Display moves with the Viewer and everything (Display and Ground) is translated to place the center of the display back to the Origin. This is also true if the Viewer were to change the Boresight.

To find the limits of the viewing triangle, **dsy** is the distance from the origin to the top of the screen point (e1x, e1y) and has a length = $[\sqrt{[e1x^2 + e1y^2]}$

It can be stated as:

$$e1x = -Yv0/\sqrt{[Xv0^2 + Yv0^2]}$$
$$e1y = Xv0/\sqrt{[Xv0^2 + Yv0^2]}$$

$$e1x^2 = Yv0^2/ [Xv0^2 + Yv0^2]$$
$$e1y^2 = Xv0^2/ [Xv0^2 + Yv0^2]$$

$$dsy^2 = Yv0^2/ \sqrt{\ } [Xv0^2 + Yv0^2] + Xv0/ \sqrt{\ } [Xv0^2 + Yv0^2]$$

$$dsy = Yv0/\sqrt{\ } [Xv0^2 + Yv0^2] + Xv0/ \sqrt{\ } [Xv0^2 + Yv0^2]$$

$$dsy = \frac{(Yv0 + Xv0)}{\sqrt{[Xv0^2 + Yv0^2]}}$$

Therefore, the limit for the top of the display is:

$$w = \frac{dsy}{\sqrt{[1+ (Yv0/Xv0)^2]}} = \frac{(Yv0 + Xv0)}{\sqrt{[Xv0^2 + Yv0^2]} * \sqrt{[1+ (Yv0/Xv0)^2]}}$$

The limit for the bottom of the display is:

$$w = \frac{dsy}{\sqrt{[1+ (Yv0/Xv0)^2]}} = \frac{(-Yv0 - Xv0)}{\sqrt{[Xv0^2 + Yv0^2]} * \sqrt{[1+ (Yv0/Xv0)^2]}}$$

Plug them into:

$$Ay = \frac{[-Yv0/Xv0 * w - Yv0] * [Ax0 - Xv0] + Yv0}{(w - Xv0)}$$

The results are shown graphically in Figure M5

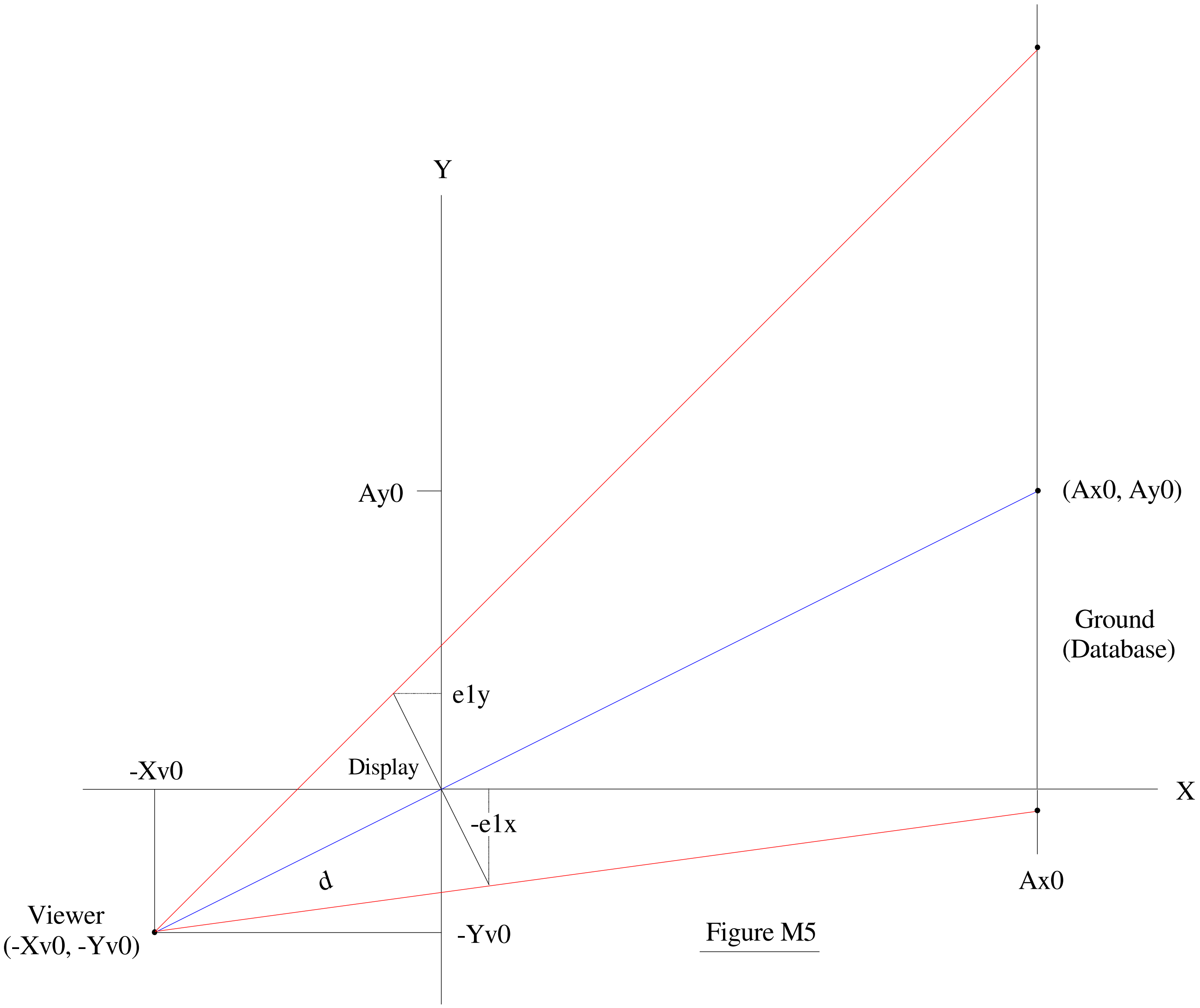

Figure M5

This is for a 2D World. For a 3D World add an axis that is normal to the X and Y axes. Call it Z.

The Ground is still perpendicular to the X axis so we will make the Z axis a clone of the Y Axis.

$$\mathbf{Az} = \frac{[-\mathbf{Zv0/Xv0} * \mathbf{w} \;-\; \mathrm{Zv0}] \;*\; [\;\mathbf{Ax0} \;-\; \mathbf{Xv0}\;] \;+\; \mathbf{Zv0}}{(\mathbf{w} \;-\; \mathrm{Xv0})}$$

[Equation M2]

Where $\mathbf{w = dsz / \sqrt{}[1+ (Zv0/Xv0)^2]}$

This puts the axes in a non-standard orientation with the X Axis Up, the Y Axis to the Right, and the Z Axis into the screen.

The Axes can all be swapped around to suit the Viewer's preferences.

This method gives us the ability to scan the display, producing the memory addresses that correspond to each pixel on the display. Each memory address can contain a color, or it can contain an elevation, or it can contain both.

But there are some problems if the memory contains elevations.

1. An elevation might not project onto the pixel that produced the address unless we are using orthographic projection. And with a perspective projection there might be several elevations that map to the same pixel.

2. If we use this method only for determining the viewing pyramid we might miss elevations beyond the border that should be displayed because of their elevation.

3. A subtle error is that, for a 3D world, making the display perpendicular to each axis does not give us the ability to roll the Viewer although it does work for pitch and yaw.

One solution is to rotate the memory coordinates to account for Roll.

$$X' = X*COS(a) - Y*SIN(a)$$

$$Y' = X*SIN(a) + Y*COS(a)$$

The other might be to go back to using the **e** vectors for the Display.

The e vectors for a 3D world would be:

$$e1(x, y, z)$$

and

$$e2(x, y, z)$$

The equations might be as follows:

$$Ay = \frac{[(e1y/e1x) * w \ - Yv0] \ * [\ Ax0 \ - Xv0\ ] \ + Yv0}{(w \ - Xv0)}$$

Where $w = dsy / \sqrt{[1+(e1y/e1x)^2]}$ [Equation M3]

$$Az = \frac{[(e2y0/e2x0) * w \ - Zv0] \ * [\ Ax0 \ - Xv0\ ] \ + Zv0}{(w \ - Xv0)}$$

Where $w = dsz / \sqrt{[1+(e2y/e2x)^2]}$

Except, it’s not going to work because the **e** vectors have no **z** value.

Somehow we have not accounted for it in either axis. It’s like we deliberately set it to zero in order to make the calculations easier, or maybe to even make them possible.

Is that what Möller did, at the expense of having the system responsive to Roll angles?

Jed Margolin
VC Highlands, NV
November 8, 2009

Reference

Synthetic Vision for Enhancing Poor Visibility Flight Operations
H. Möller, Research Assistant and G. Sachs, Director, *Institute of Flight Mechanics and Flight Control, Technische Universität Munchen;* Paper From: *IEEE AES Systems Magazine. March 1994, pages 27 - 33*

.end