3-D Primer: Making stereo views from graphics data

(This is where the funny glasses come in handy.)

By Dana Dominiak

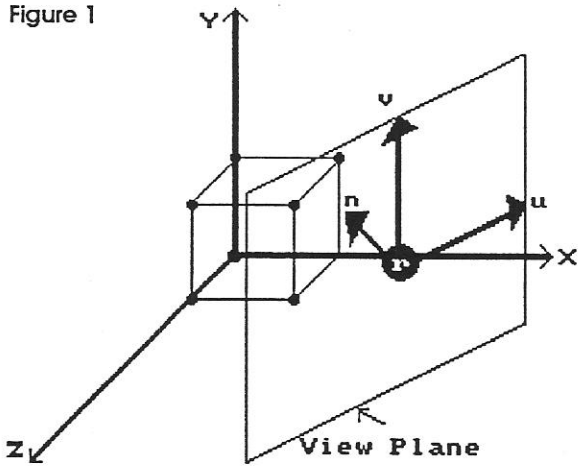
Human depth perception is achieved by the combination of two views, one from each eye, into one image. This combination is accomplished in the brain. Each eye sees a different, "flat" perspective of a scene. The view from the right eye is slightly different from the view from the left eye, since the eyes are separated by more than one inch. Even though these two views are very similar, the different information they contain is enough to distinguish between near and far objects. No depth perception is experienced without the use of both eyes. Maybe this means that if we were four-dimensional beasts living in a four-dimensional universe, we'd need three eyes to see in 4D, and really *funny* looking glasses. You've probably noticed this "stereo.separation" from your own eyes many times.

The more an object's position changes from the left eye's view to the right eye's view, the closer the object is. Creatures--like birds--whose eyes are separated too far apart to see the same image with each eye, do not see with stereo vision.

In order to create stereo imagery on a computer screen, each eye must see a different view of the same scene. The red/blue glasses are the most economical means of achieving this goal, although other methods, such a liquid-crystal "shutter" glasses, will also work fine, though at higher cost. The red/blue glasses use a red filter for one eye (usually the left) and a blue filter for the other eye. The idea is that the red lens will filter out a red image (so you can't see it with that eye). Likewise, the blue lens filters out the blue image, and only the other eye (red lens) can see it. The only other technicality is: What if both images overlap? In that case, both eyes should be able to see it. So, overlaping parts of an image should be drawn with a color that BOTH lenses will see, and not filter out, such as a nice grey.

So we now have a means for separating a stereo image for each eye, but how do we create a stereo rendition of some 3D object?

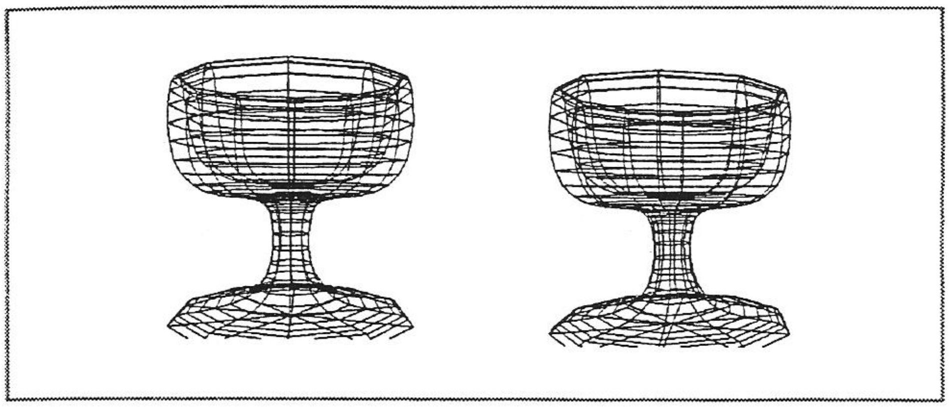
Figure 1



Mathematics of Projection (Eek!)

Data for a 3D object (Information on storage of 3D data, also in this issue), which is represented in world coordinates (X-Y-Z), must be translated into viewing (u-v-n) coordinates by means of mathematical translation. (The letters u, v, and n are commonly used to name coordinate axes, to distinguish between sets of data under consideration.) Refer to *Figure 1*. The cube is shown sitting in the world coordinate system, which is typically described as X-Y-Z. To the right of the cube is a view plane, whose coordinate system is usually described as u-v-n. A perspective view of the cube must be projected onto this view plane.

The view plane can "move" throughout the scene, tilting or moving in any direction. It is the "window" into the world coordinate system through which your eye looks. You can think of it as your eye's local coordinate system, where the u, v, and n axes correspond to x, y, and z axes. The u and v axes lie within the view plane, however, the n axis lies "normal" (i.e. at a 90-degree angle, or perpendicular) to the view plane. A view reference point r describes the center of the view plane. Unit vectors, u, v, and n describe the view plane axes. Both the view reference point and the vector u, v, and n are represented in world



3D optical illusion

The illustrations at left are views of the 3D data in Goblet.3D, from this issue's Issue Disk. You can see the goblet in 3D by holding the magazine a foot or so from your eye and staring intently at the space *between* the illustrations. Relax your eyes, as though looking into the distance, rather than focusing on the paper in front of you. After a while, you'll see an aerial image appear in full 3D.

coordinates. Study *Figure 1*, looking at the vectors r, u, v, and n.

The normal vector can be found by calculating the *cross product* of u and v. Therefore,

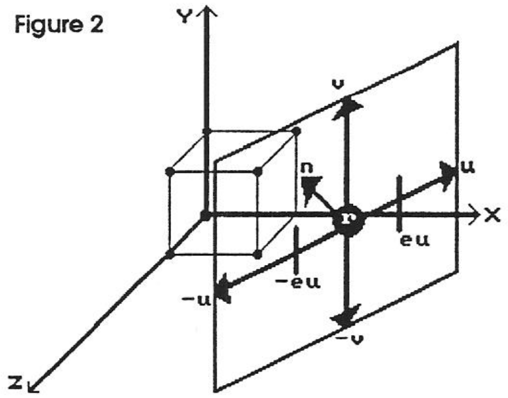
```
n = v x u
```

or (remember "Determinants" from Linear Algebra? No?):

$$n = \begin{vmatrix} v.x & v.y & v.z \\ u.x & u.y & u.z \end{vmatrix}$$

```
n.x = v.y * u.z - v.z * u.y
n.y = v.z * u.x - v.x * u.z
n.z = v.x * u.y - v.y * u.x
```

Figure 2



Now that we have our coordinate systems set up, we must account for two separate views--remember, one for each eye. We can place the right eye (eu) along the positive u axis (see *Figure 2*). Similarly, we can place the left eye (-eu) in the opposite direction along the negative x axis. Notice that eu is now given in *view plane* coordinates. Also, the eyes should lie back a little from the view plane, at -n.

Finally, we can project the cube onto the view plane with

$$u = \frac{en * pu - eu * pn}{en - pn}$$

$$v = \frac{en * pv - ev * pn}{en - pn}$$

where a point in viewing coordinates p = (pu, pv, pn).

Converting viewpoints

To convert a point to viewing coordinates, simply do the following:

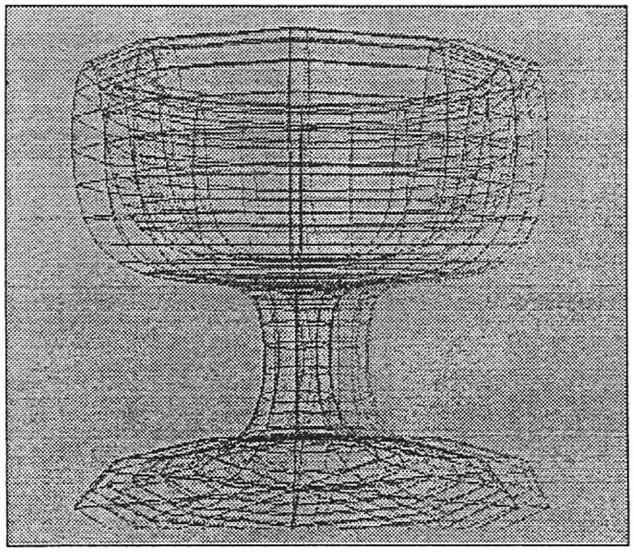
```
pu = ((world_point.x - r.x) *
    u.x) +
      ((world_point.y - r.y) *
    u.y) +
      ((world_point.z - r.z) *
    u.z)
```

Similar equations are obvious for pv and pn.

The other eye

The above projection was for the *right eye only*. (That almost sounded like the title of a James Bond movie, didn't it?). That projection should be drawn in red, so the blue filter of the right eye will see it, but the red filter on the left eye will block it out. Next, the above projection must be recalculated with -eu, and drawn in blue for the left eye. If the images overlap, these pixels should be illuminated grey so that "holes" don't appear in one eye's view.

Surprisingly, that's all there is to creating a stereo view of 3D data. Nothing like a little mathematics to get the blood to flowing, too.


Left Side, plus Right Side, equals 3D. (But it would have to be printed in red/blue.)