

## Shadows

Every possible projection of three-dimensional space to three-dimensional space can be achieved with a suitable  $4 \times 4$  invertible matrix and homogeneous coordinates. If the matrix isn't invertible but has rank 3, it projects three-dimensional space onto a two-dimensional plane. Every such possible projection can be achieved with a suitable rank-3  $4 \times 4$  matrix. To find the shadow of an arbitrary object on an arbitrary plane from an arbitrary light source (possibly at infinity), you need to find a matrix representing that projection, multiply it on the matrix stack, and draw the object in the shadow color. Keep in mind that you need to project onto each plane that you're calling the "ground."

As a simple illustration, assume the light is at the origin, and the equation of the ground plane is  $ax + by + cz + d = 0$ . Given a vertex  $S = (sx, sy, sz, 1)$ , the line from the light through  $S$  includes all points  $\alpha S$ , where  $\alpha$  is an arbitrary real number. The point where this line intersects the plane occurs when

$$\alpha(a \cdot sx + b \cdot sy + c \cdot sz) + d = 0,$$

so

$$\alpha = -d / (a \cdot sx + b \cdot sy + c \cdot sz).$$

Plugging this back into the line, we get

$$-d(sx, sy, sz) / (a \cdot sx + b \cdot sy + c \cdot sz)$$

for the point of intersection.

The matrix that maps  $S$  to this point for every  $S$  is

$$\begin{bmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & -d & 0 \\ a & b & c & 0 \end{bmatrix}$$

This matrix can be used if you first translate the world so that the light is at the origin.

If the light is from an infinite source, all you have is a point  $S$  and a direction  $D = (dx, dy, dz)$ . Points along the line are given by

$$S + \alpha D$$

Proceeding as before, the intersection of this line with the plane is given by

$$a(sx + \alpha \cdot dx) + b(sy + \alpha \cdot dy) + c(sz + \alpha \cdot dz) + d = 0$$

Solving for  $\alpha$ , plugging that back into the equation for a line, and then determining a projection matrix gives

$$\begin{bmatrix} b \times dy + c \times dz & -b \times dx & -c \times dx & -d \times dx \\ -a \times dy & a \times dx + c \times dz & -c \times dy & -d \times dy \\ -a \times dz & -b \times dz & a \times dx + b \times dy & -d \times dz \\ 0 & 0 & 0 & a \times dx + b \times dy + c \times dz \end{bmatrix}$$

This matrix works given the plane and an arbitrary direction vector. There's no need to translate anything first. (See Chapter 3 and Appendix F.)

## Hidden-Line Removal

If you want to draw a wireframe object with hidden lines removed, one approach is to draw the outlines using lines and then fill the interiors of the polygons making up the surface with polygons having the background color. With depth-buffering enabled, this interior fill covers any outlines that would be obscured by faces closer to the eye. This method would work, except that there's no guarantee that the interior of the object falls entirely inside the polygon's outline; in fact, it might overlap it in various places.

There's an easy, two-pass solution using either polygon offset or the stencil buffer. Polygon offset is usually the preferred technique, since polygon offset is almost always faster than stencil buffer. Both methods are described here, so you can see how both approaches to the problem work.

### Hidden-Line Removal with Polygon Offset

To use polygon offset to accomplish hidden-line removal, the object is drawn twice. The highlighted edges are drawn in the foreground color, using filled polygons but with the polygon mode `GL_LINE` to rasterize the polygons as wireframe edges. Then the filled polygons are drawn with the default polygon mode, which fills the interior of the wireframe, and with enough polygon offset to nudge the filled polygons a little farther from the