Angel and Shreiner: Interactive Computer Graphics, Sixth Edition

Chapter 4 Solutions

4.1 Eclipses (both solar and lunar) are good examples of the projection of an object (the moon or the earth) onto a nonplanar surface. Any time a shadow is created on curved surface, there is a nonplanar projection. All the maps in an atlas are examples of the use of curved projectors. If the projectors were not curved we could not project the entire surface of a spherical object (the Earth) onto a rectangle.

4.2 Suppose that the axis of the plane is its z direction and up is the y direction. In the airplane's coordinate system, the roll, pitch and yaw correspond to rotations about the z, x and y axes respectively. Thus we can control the orientation relative the origin by a rotation of the form $\mathbf{R}_z(roll)\mathbf{R}_x(pitch)\mathbf{R}_y(yaw)$. We must also do a translation to move the airplane to its desired location.

4.3 Suppose that we want the view of the Earth rotating about the sun. Before we draw the earth, we must rotate the Earth which is a rotation about the y axis. Next we translate the Earth away from the origin. Finally we do another rotation about the y axis to position the Earth in its desired location along its orbit. There are a number of interesting variants of this problem such as the view from the Earth of the rest of the solar system.

4.4 This problem is very similar to the next problem. We use the VRP to translate the camera. The VPN becomes the new z direction. We can compute the new x direction by taking the cross product of the VPN and the VUP vector. The new y will be the cross product of the new z and the new x. Note that all vectors must be normalized and we must be careful of the order of the vectors in the cross product.

4.5 Yes. Any sequence of rotations is equivalent to a single rotation about a suitably chosen axis. One way to compute this rotation matrix is to form the matrix by sequence of simple rotations, such as

$$\mathbf{R} = \mathbf{R_x}\mathbf{R_y}\mathbf{R_z}.$$

The desired axis is an eigenvector of this matrix.

4.6 If we start with a translation of the COP to the origin, the remaining problem is to find the intersection of a projector from the origin to a point

(x, y, z) with the plane $ax + by + cz + d = 0$. Any point on this projector can be written as $(\alpha x, \alpha y, \alpha z)$. Substituting into the equation of the plane we find $\alpha = \frac{-d}{ax+by+cz}$ and the point of intersection is $\frac{-d}{ax+by+cz}(x, y, z)$. We can use the homogeneous coordinate form to compute the denominator as the w term, yielding the matrix

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

4.7 Consider the line determined by the points (x_1, y_1, z_1) and (x_2, y_2, z_2). Any point along can be written parametrically as $(\alpha x_1 + (1 - \alpha)x_2, \alpha y_1 + (1 - \alpha)y_2, \alpha z_1 + (1 - \alpha)z_2)$. Consider the simple projection of this point $\frac{1}{d(\alpha z_1+(1-\alpha)z_2)}(\alpha x_1 + (1 - \alpha)x_2, \alpha y_1 + (1 - \alpha)y_2)$ which is of the form $f(\alpha)(\alpha x_1 + (1 - \alpha)x_2, \alpha y_1 + (1 - \alpha)y_2)$. This form describes a line because the slope is constant. Note that the function $f(\alpha)$ implies that we trace out the line at a nonlinear rate as α increases from 0 to 1.

4.8 As we project points on one side of the projection plane, getting closer and closer to where the line segment intersects the projection plane, the projected points move off to infinity. If we do the same thing for points on the other side of the projection plane, the projected points move off to infinity in the opposite direction. Hence, we can conclude that the projection of a line segment whose endpoints are on opposite sides of the projection plane is a line segment that goes off to and returns from infinity.

4.9 The specification used in many graphics texts is to use the angles the projector makes with x,z and y, z planes, i.e the angles defined by the projection of a projector by a top view and a side view.
Another approach is to specify the foreshortening of one or two sides of a cube aligned with the axes.

4.10 If the camera is at the origin, we can assume that the angle of projection is determined by the line from the origin to the center of the window $((x_{min} + x_{max})/2, (y_{min} + y_{max})/2)$. Thus, the two required cotangents for the projection matrix are

$$\cot\theta = \frac{2z_{near}}{x_{min} + x_{max}}$$

$$\cot \phi = \frac{2z_{near}}{x_{min} + x_{max}}$$

4.11 The CORE system used this approach. Retained objects were kept in distorted form. Any transformation to any object that was defined with other than an orthographic view transformed the distorted object and the orthographic projection of the transformed distorted object was incorrect.

4.12 All are changed by a translation matrix $\mathbf{T}(0, 0, -d)$.

4.15 If we use $\theta = \phi = 45$, we obtain the projection matrix

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.17 All the points on the projection of the point $(x.y, z)$ in the direction $d_x, d_y, d_z)$ are of the form $(x + \alpha d_x, y + \alpha d_y, z + \alpha d_z)$. Thus the shadow of the point (x, y, z) is found by determining the α for which the line intersects the plane, that is

$$ax_s + by_s + cz_s = d$$

Substituting and solving, we find

$$\alpha = \frac{d - ax - by - cz}{ad_x + bd_y + cd_z}.$$

However, what we want is a projection matrix, Using this value of α we find

$$x_s = z + \alpha d_x = \frac{x(bd_y + cd_x) - d_x(d - by - cz)}{ad_x + bd_y + cd_z}$$

with similar equations for y_s and z_s. These results can be computed by multiplying the homogeneous coordinate point $(x, y, z, 1)$ by the projection matrix

$$\mathbf{M} = \begin{bmatrix} bd_y + cd_z & -bd_x & -cd_x & -dd_x \\ -ad_y & ad_x + cd_z & -cd_y & -dd_y \\ -ad_z & -bd_z & ad_x + bd_y & -dd_z \\ 0 & 0 & 0 & ad_x + bd_y + cd_z \end{bmatrix}.$$

4.21 Suppose that the average of the two eye positions is at (x, y, z) and the viewer is looking at the origin. We could form the images using the `LookAt` function twice, that is

```
LookAt(x-dx/2, y, z, 0, 0, 0, 0, 1, 0);
/* draw scene here */
/* swap buffers and clear */
LookAt(x+dx/2, y, z, 0, 0, 0, 0, 1, 0);
/* draw scene again */
/* swap buffers and clear */
```

4.22 One approach is to use a repeated vertex at the end of each line, this creating a degenerate triangle with no area. Another is to make the line segment going across either transparent or in the background color. Another approach is to use multiple strips, one for each traverse across.

4.23 Probably the simplest approach would be the one in the previous problem of repeating vertices at the end of each row to create a degenerate polygon.