Angel and Shreiner: Interactive Computer Graphics, Eighth Edition

Chapter 1 Solutions

1.1 The main advantage of the pipeline is that each primitive can be processed independently. Not only does this architecture lead to fast performance, it reduces memory requirements because we need not keep all objects available. The main disadvantage is that we cannot handle most global effects such as shadows, reflections, and blending in a physically correct manner.

1.2 If we use lines of constant longitude and lines of constant latitude, their intersections define a set of quadrilaterals. If we draw diagonals for each quadrilateral, we get a set of triangles. However, at the poles all the curves of constant longitude meet and we get triangles rather than quadrilaterals.

1.3 We derive this algorithm later in Chapter 6. First, we can form the tetrahedron by finding four equally spaced points on a unit sphere centered at the origin. One approach is to start with one point on the z axis $(0, 0, 1)$. We then can place the other three points in a plane of constant z. One of these three points can be placed on the y axis. To satisfy the requirement that the points be equidistant, the point must be at $(0, 2\sqrt{2}/3, -1/3)$. The other two can be found by symmetry to be at $(-\sqrt{6}/3, -\sqrt{2}/3, -1/3)$ and $(\sqrt{6}/3, -\sqrt{2}/3, -1/3)$.

We can subdivide each face of the tetrahedron into four equilateral triangles by bisecting the sides and connecting the bisectors. However, the bisectors of the sides are not on the unit circle so we must push these points out to the unit circle by scaling the values. We can continue this process recursively on each of the triangles created by the bisection process.

1.4 Suppose that the line segment is between the points (x_1, y_1) and (x_2, y_2) We can use the endpoints of the line segment to determine the slope and y intercept of a line of which the segment is part, i.e.

$$y = mx + h = \frac{y_2 - y_1}{x_2 - x_1}x + y_1 - \frac{y_2 - y_1}{x_2 - x_1}x_1.$$

Note that we can deal with horizontal and vertical line segments as special cases. We can find the intersections with the sides of the window by substituting $y = y_{max}$, $y = y_{min}$, $x = x_{max}$, and $x = x_{min}$ (the equations for the sides of the window) into the above equation. We can check the

locations of the points of intersection to determine if they are on the line segment or only on the line of which it is part.

1.5 In Exercise 1.4, we saw that we could intersect the line of which the line segment is part independently against each of the sides of the window. We could do this process iteratively, each time shortening the line segment if it intersects one side of the window.

1.6 Here we have to intersect the line segment against the polygons that determine the sides of the window. Each of these polygons is part of a plane. The equation of each plane is of the form

$$ax + by + cz + d = 0,$$

where the coefficients a, b, c, and d can be determined from the vertices of the parallelepiped. The line segment is either parallel to a particular plane or intersects it in exactly one place that can be determined from the equation of the line. For a line passing through the two three–dimensional points (x_1, y_1, z_1) and (x_2, y_2, z_2), we can write its equation in parametric form as

$$x(t) = (1 - t)x_1 + tx_2,$$
$$y(t) = (1 - t)y_1 + ty_2,$$
$$z(t) = (1 - t)z_1 + tz_2.$$

We can substitute this equation in the equation for a plane and determine t for the point of intersection.

1.7 In a one–point perspective, two faces of the cube is parallel to the projection plane, while in a two–point perspective only the edges of the cube in one direction are parallel to the projection. In the general case of a three–point perspective there are three vanishing points and none of the edges of the cube are parallel to the projection plane.

1.8 We have to process 1280 x 1024 x 72 pixels/sec. If we process each successively, there is only about 10 nanoseconds to process each. For a 480 x 640 interlaced display operating at 60 Hz we must process only 480 x 640 x 30 pixels/sec which gives us about 109 nanoseconds to process each pixel.

1.9 Each frame for a 480 x 640 pixel video display contains only about 300k pixels whereas the 2000 x 3000 pixel movie frame has 6M pixels, or about 18 times as many as the video display. Thus, it can take 18 times as much time to render each frame if there is a lot of pixel-level calculations.

1.10 Good examples for this problem are architecture, mechanical parts design, electrical circuit design and a paint program.

1.11 A 1024 x 1280 display has a 4 to 5 aspect ratio. Hence, if the diagonal is 50 cm and we want square pixels, the screen must be approximately 31 cm x 39 mm. Each pixel is then about 0.3 mm on each side. A smooth display will require about 3 triads for each pixel, and thus the triads are about 0.1 mm apart. Finally if the shadow mask is halfway between the screen and electron guns, the shadow mask spacing is half the triad spacing.