International Tables for Crystallography (2006). Vol. B, Section 3.3.1.5.6, p. 376.

of partially obscured atoms near the front to be determined before they can be painted or, alternatively, every pixel must be tested before loading to see if it is already loaded. Not only does this approach give a uniform rendering over the whole area of one atom, it also gives a boundary between overlapping atoms with almost equal z values which completes the circle of the nearer atom, though it should be an arc of an ellipse when the atoms are drawn with radii exceeding their covalent radii.

Greater realism is achieved by establishing a z buffer, which is an additional area of memory with one word per pixel, in which is stored the z value of the currently loaded feature in each pixel. Treatments which take account of the sphericity are then possible and correct arcs of intersection for interpenetrating spheres and more complicated entities arise naturally through loading a colour value into a pixel only if the z coordinate is less than that of the currently loaded value. This z buffer and the associated x, y coordinates should be in picture space or screen space rather than display space since only after the application of perspective can points with the same x/w and y/w coordinates obscure one another.

It is usual in such systems to vary the intensity of colour within one atom by darkening it towards the circumference on the basis of the z coordinate. Some systems augment this impression of sphericity by highlighting. The simplest form of highlighting is an extension of the uniform disc treatment in which additional, brighter discs, possibly off centre, are associated with each atom. More general highlighting (Phong, 1975) is computed from four unit vectors, these being the normal to the surface, the direction to a light source, the direction to the viewer and the normalized vector sum of these last two. Intensity levels may then be set as the sum of three terms: a constant, a term proportional to the scalar product of the first two vectors (if positive) and a term proportional to a high power of the scalar product of the first and last vectors; the higher the power the glossier the surface appears to be. This final term normally adds a white term, rather than the surface colour, supposing the light source to be white.

Shadows may also be rendered to give even greater realism. In addition to the z buffer and (x, y) frame buffer a second z buffer for z' values associated with x' and y' is also required. These coordinates are then related by $x' = x + \alpha z$, $y' = y + \beta z$, $z' = z$. The second buffer is a ray buffer since $x'y'$ are the coordinates with which an illuminating ray passing through (xyz) passes through the $z = 0$ plane, and z', stored at x', y', records the depth at which this ray encounters material. Thus any two pixels $(x_1y_1z_1)$ and $(x_2y_2z_2)$ are on the same illuminating ray if their x' and y' values are equal and the one with smaller z' shadows the other. Processing a pixel at $(x_1y_1z_1)$ therefore involves first determining its visibility on the basis of the z buffer, as before, then, whether or not it is visible, setting $z'_1 = z_1$ and considering the value of z' currently stored at $x'y'$, which we call z'_2.

If $z'_1 < z'_2$ then $x_1y_1z_1$ is in light and must be loaded accordingly. From z'_2 we find the previously processed pixel $(x_2y_2z_2)$ which is now in shade and which was in light when originally processed, so that the colour value stored at x_2y_2 needs to be altered *unless* the pixel at x_2y_2 is now $(x_2y_2z_3)$ with $z_3 < z_2$, in which case the pixel $(x_2y_2z_2)$ which has now become shadowed by $(x_1y_1z_1)$ has, in the meantime, been obscured by $(x_2y_2z_3)$ which is not shadowed by $(x_1y_1z_1)$ and no change is therefore needed. In either event z'_1 then replaces z'_2.

If $z'_1 > z'_2$ then $(x_1y_1z_1)$, if visible, is in shade and must be coloured accordingly, and in this case z'_2 is not superseded.

This shadowing scheme corresponds to illumination by a light source at infinity in picture space or, equivalently, with a z coordinate equal to that of the eye in display space. For its implementation x, y and z may be in any convenient coordinate system, *e.g.* pixel addresses, but if x and y are expressed with the range -1 to 1 and z with the range 0 to 1 corresponding to the window then they may be identified as the quantities $x/w, y/w$ and z/w of picture space (Section 3.3.1.3.1).

If, in the notation of Section 3.3.1.3.5, the light source is placed at (P, Q, E, V) in display space and a ray leaves it in the direction (p, q, r, V) then

$$x' = \frac{p}{r} \cdot \frac{2(S-E)}{(R-L)} + \frac{2(S-E)(P-C)}{(N-E)(R-L)} + \frac{2C-R-L}{R-L},$$

which varies only with beam direction,

$$\alpha = \frac{2(S-E)(F-N)(P-C)}{(F-E)(N-E)(R-L)}$$

and similarly for y' and β.

3.3.1.5.6. *Advanced hidden-line and hidden-surface algorithms*

Hidden surfaces may be handled quite generally with the z-buffer technique described in the previous section but this technique becomes very inefficient with very complicated scenes. Faster techniques have been developed to handle computations in real time (*e.g.* 25 frames s^{-1}) on raster machines when both the viewpoint and parts of the environment are moving and substantial complexity is required. These techniques generally represent surfaces by a number of points in the surface, connected by lines to form panels. Many algorithms require these panels to be planar and some require them to be triangular. Of those that permit polygonal panels, most require the polygons to be convex with no re-entrant angles. Yet others are limited to cases where the objects themselves are convex. Some can handle interpenetrating surfaces, others exclude these. Some make enormous gains in efficiency if the objects in the scene are separable by the insertion of planes between them and degrade to lower efficiency if required, for example, to draw a chain. Some are especially suited to vector machines and others to raster machines, the latter capitalizing on the finite resolution of such systems. In all of these the basic entities for consideration are entire panels or edges, and in some cases vertices, point-by-point treatment of the entire surface being avoided until after all decisions are made concerning what is or is not visible.

All of these algorithms strive to derive economies from the notion of 'coherence'. The fact that, in a cine context, one frame is likely to be similar to the previous frame is referred to as 'frame coherence'. In raster scans line coherence also exists, and other kinds of coherence can also be identified. The presence of any form of coherence may enable the computation to be concerned primarily with changes in the situation, rather than with the totality of the situation so that, for example, computation is required where one edge crosses in front of another, but only trivial actions are involved so long as scan lines encounter the projections of edges in the same order.

The choice of technique from among many possibilities may even depend on the viewpoint if the scene has a statistical anisotropy. For example, the depiction of a city seen from a viewpoint near ground level involves many hidden surfaces. Distant buildings may be hidden many times over. The same scene depicted from an aerial viewpoint shows many more surfaces and fewer overlaps. This difference may swing the balance of advantage between an algorithm which sorts first on z or one which leaves that till last.

These advanced techniques have, so far, found little application in crystallography, but this may change. Ten such techniques are critically reviewed and compared by Sutherland *et al.* (1974), and three of these are described in detail by Newman & Sproull (1973).

references