Atoms with shadows — an area based algorithm for cast shadows on space-filling molecular models

Michael Gwilliam and Nelson Max

University of California, Davis, Department of Applied Science and Lawrence Livermore National Laboratory, Livermore, CA, USA

An area-based algorithm for space-filling molecular models, which computes the visible regions of each atom sphere at floating-point precision, has been used to produce hard-edged shadows. The algorithm is repeated from the point of view of the light source in the YZ plane, and the visible regions are then transformed into the illuminated regions in the final view. The cost of the visibility calculations is independent of the resolution of the raster.

Keywords: computer graphics; shadowing; shading; space-filling molecular models; scanline algorithm

VISIBILITY CALCULATIONS FOR MOLECULAR MODELS

Space-filling molecular models are built from intersecting spheres with radii equal to the van der Waals radii of the atoms they represent. When two nonbonded atoms are at their equilibrium distance under the van der Waals force, their spheres just touch. These space-filling models are useful for studying molecular interactions and visualizing molecular structure. They were originally built from plastic truncated spheres, but there are now several computer algorithms for rendering them in color on raster video terminals. Among these are z-buffer, ray-tracing and area-based algorithms.

Duff[1] has applied an antialiased z-buffer algorithm that precomputed depths and shading for standard sized spheres. Pique[2] also uses a z-buffer algorithm, but computes the depth and shading with forward differences, using a table lookup for the square root in the depth formula.

Ken Knowlton[3] has also applied the back-to-front "painter's" algorithm to scenes consisting of spheres, and Tom Porter[4] has used a scan line version of the z-buffer to save memory. For further discussion of these and other algorithms, see Max.[5]

Address reprint requests to Michael Gwilliam, at his present address: Computer Graphics Laboratory, New York Institute of Technology, Old Westbury, NY 11568.

Received 15 August 1988; accepted 2 December 1988

Ray tracing can generate spectacular looking scenes showing multiple reflections of one surface in another. It is particularly easy to implement for scenes consisting only of spheres because of the simplicity of the ray/sphere intersection calculations. However, for large molecules it is inefficient to test the ray against every sphere, so methods of reducing the set of tested spheres are necessary. Such methods are discussed in Fujimoto,[6] Glasner,[7] and Avro and Kirk.[8]

In determining which sphere is visible at a pixel, both the ray-tracing and z-buffer algorithms require time proportional to the number of pixels. Even more time will be required for antialiasing, which is usually done by calculating at a higher resolution and then averaging down to the final raster. In contrast, area-based algorithms calculate the screen regions (which are visible portions of each object) to arbitrary precision, which is usually limited to the floating-point precision of the computer. These regions can later be rendered at any desired resolution. The final shading cost will be resolution dependent, but the visibility calculation will not. Instead, it will depend on the number of atoms in the input molecule and the complexity of the image (i.e., the number of visible regions in screen space). In addition, the precise outlines of the visible regions provide the information necessary for accurate antialiasing.

In 1976, Ken Knowlton and Lorinda Cherry[9] published an area-based algorithm for molecular models built from spheres and cylinders. It could do the visibility calculations for ball-and-stick models, as well as for space-filling models. The visible regions on such scenes are bounded by circles, ellipses and straight lines. As explained below, the ellipses were approximated by circles, so the regions were particularly easy to define. The plan was to color the visible regions with a minicomputer, which needed no knowledge of the 3D model that generated the regions. Originally, a constant flat color was used, but later Max[10] used fixed-point arithmetic in microcode to generate the shading and highlights in conjunction with hardware lookup tables in the film recorder.

SHADOW CALCULATIONS

Shadows provide extra visual cues for understanding the 3D structure of a molecule rendered in a raster image. One method of computing them is to repeat the visibility calculations from the point of view of the light source. The visible regions must be transformed back to the final view, where they become the illuminated regions, while the rest remains in shadow. This method works for the z-buffer algorithm since the x,y and z coordinates of each illuminated point can be found from the z-buffer, as discussed in Williams[11] and Reeves *et al.*[12] Attention must be paid to avoid aliasing when the discrete (x, y, z) points in the light source view are transformed to the final image. The light source view method also applies to area-based algorithms, as was first shown by Weiler and Atherton[13] for scenes modeled with polygons. Their "polygons in, polygons out" technique makes shadows easy, since the illuminated polygons transform back to subpolygons of the original input data.

For spheres, things are not as simple, since illuminated regions are bounded by fourth-degree curves in the plane of the final image, as we will show later. We will also explain how we deal with this problem by transforming one scan line at a time.

Ray-tracing algorithms find shadows by tracing a ray from each visible surface point toward the light source and testing whether the ray encounters any other surfaces before it gets there. For both the z-buffer and ray tracing, the illumination computation time is proportional to the number of pixels, which is the square of the resolution. For an area-based algorithm the shadow calculations can, in principle, be resolution independent. However, since our algorithm for spheres transforms one scan line of data at a time, the transformation calculations grow linearly with the resolution.

Max[14] gives a related shadow algorithm in which the fraction of the area of the sphere that is visible in the light source is used to multiply all shading values on the sphere, except for the "ambient" illumination that persists in the shadows. That approximation gives soft, penumbra-like shadows, instead of accurate, hard-edged cast shadows, but the cost for the shadows is again independent of the resolution.

The algorithm presented here uses the aforementioned method of Knowlton and Cherry twice to produce one view from the observer and one from the light source which is constrained to be at infinity in the YZ plane. The two databases are compared, atom by atom, to determine which parts of the atom, visible to the observer, are also visible to the light source. The regions visible to both are shaded as specular reflecting spheres, and the regions visible only to the observer are diffusely shaded as if there were a secondary, very dim, light source at the viewer's eye. A third, even more subtle, light source is present as a diffuse light source that sets a minimum light level for every part of the atom. The methods for computing each of these regions are detailed below.

DESCRIPTION OF VISIBLE REGIONS

Figure 1 shows a sphere in an extreme wide angle perspective projection. The viewpoint V is a distance of one unit from the picture plane center W. The sphere's center $C = (x,y,z)$ projects to a point $D = (x', y')$ on the picture plane with $x' = x/z$ and $y' = y/z$. A cone of rays tangent to the sphere meet it in a profile circle S', with center E along the cone axis VC. This cone intersects the picture plane in an ellipse T. For simplicity, Knowlton and Cherry[9] approximated this ellipse by a circle U of radius $r' = r/z'$. For the narrow angle view usual in application, this is a very good approximation. It greatly simplifies the visible surface calculation for two reasons. First, a circle is defined by three parameters, while an ellipse takes five parameters. Second, two circles can intersect in at most two points, which can be found by solving a quadratic equation. On the other hand, two ellipses can intersect in up to four points, and a quartic equation must be solved to find them.

Figure 2 shows two spheres intersecting in a circle C, which lies in a plane perpendicular to the line FG joining the spheres' centers. When projected, this circle should also be an ellipse. Knowlton and Cherry approximate this ellipse by a single circular arc, causing slight corners where the intersection arc meets a sphere profile (see Figure 3). The intersection curve can be made to meet the profile smoothly if several circular arcs are used (see Max[14]). However, in practice, users do not find the slight corners disturbing, so we have used a single arc here.

Knowlton and Cherry broke up the visible regions of a sphere into pieces they called *trapezoids*, with

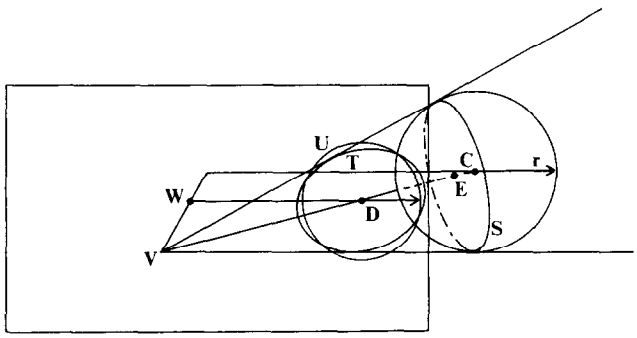

Figure 1. Approximate and exact projections of a sphere on the picture plane

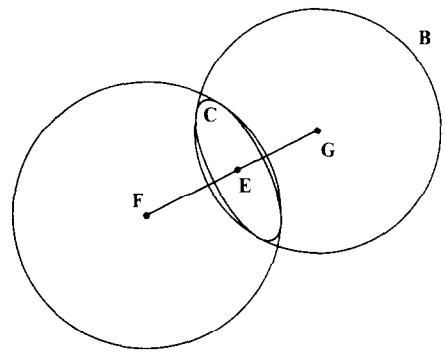

Figure 2. The circle of intersection crested by two intersecting spheres. Notice how this circle projects to an ellipse

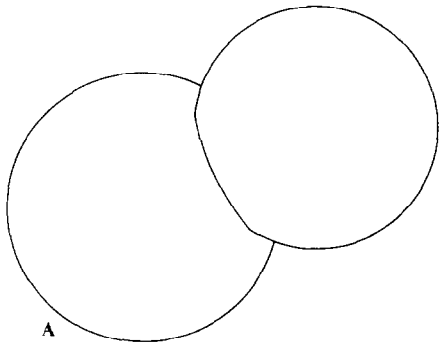

Figure 3. Elliptical projection of intersection approximated by a circle

straight vertical sides and either arcs of circles or slanted lines on the top and bottom. (Straight lines can arise if the radius of a circular arc exceeds the range of our fixed-point representation.)

The visibility calculation is performed one sphere at a time. Initially, a sphere's projection is divided up into two trapezoids by a vertical diameter, as shown in Figure 4. Each of the two "parts" has a zero length vertical side. The left "part" has a zero-length left side, and the right "part" has a zero-length right side.

In Figure 5, the volume of a second sphere intersects the first. "Mask" trapezoids are created that define the visible region of the second sphere. In the inner loop of the algorithm, the "mask" trapezoids are compared one by one with the "part" trapezoids from the first sphere, and the "part" trapezoids are deleted, modified or subdivided appropriately, as shown in Figure 5.

In Figure 6, a third sphere is added that hides, but does not intersect, the first sphere. Such a sphere can also modify the list of "part" trapezoids of the original sphere. When all spheres that can intersect or hide the original sphere have been considered, the final list of trapezoids decribes the remaining visible region. This trapezoid description of the visible region makes its successive modification easier to implement, compared with a description in terms of boundary contours. It also makes the visible regions easy to render; the trapezoids are simply rendered one by one, along vertical scan lines.

SHADOWS OF SPHERES ON A SPHERE AT THE ORIGIN

For simplicity, we will first consider the light source and the viewer to be at infinity, so that both the light source view and the final image are orthogonal projections. Consider a sphere A of radius r centered at the origin, shadowed by sphere B, as shown in Figure 7, and suppose the viewpoint is along the negative z-axis. The shadow volume cast by sphere B is a semi-infinite cylinder, defined by a quadratic polynomial in x, y and z (and a linear inequality that does not concern us here). If this quadratic equation is solved for z using the quadratic formula, we will get a solution of the form $z = L(x,y) \pm (Q(x,y))^{1/2}$ where $L(x,y)$ is a linear polynomial in x and y, and Q is a quadratic polynomial.

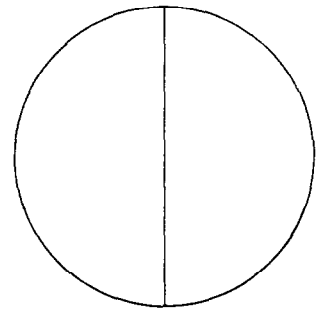

Figure 4. A sphere's initial subdivision into two trapezoids

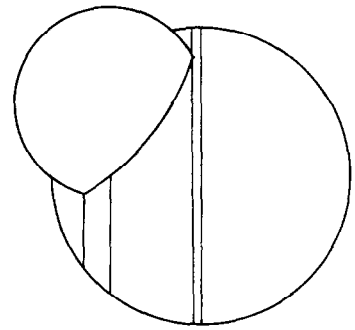

Figure 5. Additional trapezoids created by an intersecting sphere

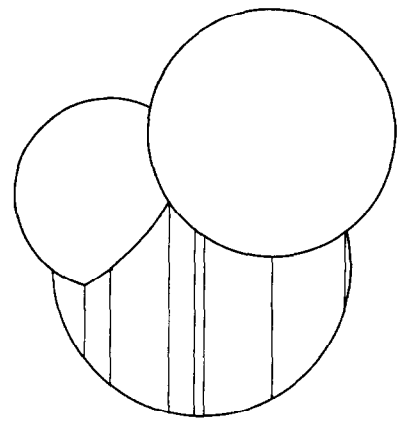

Figure 6. Additional trapezoids created by an overlapping sphere

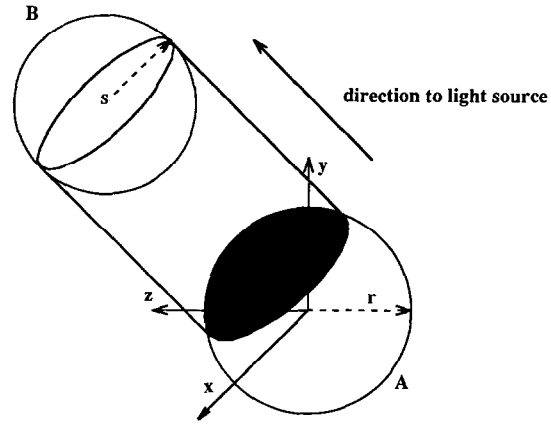

Figure 7. Cylinder of shadow cast by one sphere onto another

The equation for sphere A is $x^2+y^2+z^2 = r^2$ and substituting the above formula for z, we get

$$x^2+y^2+L(x,y)^2 \pm 2L(x,y)(Q(x,y))^{1/2}+Q(x,y)\text{-}r^2=0$$

Bringing the square root term to the right hand side, we get

$$x^2+y^2+L(x,y)^2+Q(x,y)\text{-}r^2=\mp 2L(x,y)(Q(x,y))^{1/2}$$

and squaring, we get

$$(x^2+y^2+L(x,y)^2 + Q(x,y)\text{-}r^2)^2 = 4(L(x,y))^2Q(x,y)$$

In general, this is a fourth-degree equation, so the projection in the (x,y) image plane is bounded in part by a quartic curve.

SHADOWS USING TRAPEZOIDS: THE SIMPLE CASE

Assume, as above, that the light source and the viewpoint are at infinity, and the sphere A is at the origin. In addition, assume that the light source lies in the (y,z) plane. Then a simple rotation R about the x-axis takes the (x,y) picture plane into the light source view. It is difficult to deal with shadows in the (x,y) plane, where they are bounded by quartics, but in the light source view, the illuminated regions have been described by trapezoids. So we can tell if an image pixel is illuminated by finding its z coordinate, rotating the corresponding (x,y,z) point into the light source view and checking whether it is in one of the visible trapezoids. This is time consuming, so we take advantage of coherence along vertical scan lines. The rotation R takes vertical scan lines in the (x,y) plane to parallel lines in the light source view, which are vertical in the appropriate coordinate system (x,u). For a given vertical scan line S at $x=x_0$, we find all the light source visible trapezoids intersecting $R(S)$. Each interval of intersection of such a trapezoid with $R(S)$ defines an illumination arc of the sphere. This arc is projected into S in the (x,y) image plane using the following change of coordinates.

Let V be an axis in the (y,z) plane, pointing toward the light source, let U be an axis perpendicular to X and V and let (x_0, u_1) and (x_0, u_2) be the endpoints where $R(S)$ meets the top and bottom of a light source view trapezoid.

Using the equation $x^2+u^2+v^2 = r^2$ of the sphere A in the XUV coordinate system, we find that $v_1 = (r^2 - x_0^2 - u_1^2)^{1/2}$ and $v_2 = (r^2 - x_0^2 - u_2^2)^{1/2}$, so that the points $P_1 = (x_0,u_1,v_1)$ and $P_2 = (x_0,u_2,v_2)$ bound the illuminated arc of the sphere. Then using the matrix for the rotation R about the x-axis, we transform coordinates to (x,y,z) and find $P_1 = (x_0,y_1,z_1)$ and $P_2 = (x_0,y_2,z_2)$.

If z_1 and z_2 are positive, (x_0, y_1) and (x_0,y_2) bound an illuminated interval on S. If both z_1 and z_2 are negative, the illuminated arc is on the back surface of sphere A, and it is not visible to the light source. If one of them, say z_1, is positive and the other is negative, then the visible part of the arc projects to an interval bounded by (x_0,y_1) and a point (x_0,y_0) on the profile of the sphere. In the situation shown in Figure 7, with the light coming from above, the profile point will be on the top arc, with $y_0 = (r^2-x_0^2)^{1/2}$.

The visible atoms are shaded one by one, while comparing them constantly to their counterpart as seen by the light source. If an atom is not seen by the light source, then it is in shadow and may be shaded as such. Atoms and portions of atoms that are in shadow are shaded as if there were a secondary and very dim light source at the viewer's eye. This assumption prevents unlit atoms from appearing flat, as would be the case if they were simply assigned an ambient value.

For each visible atom, its trapezoids are shaded one by one, scan line by scan line. First the scan line is painted as if it were completely in shadow and then it is brightened up wherever it is visible to the light source. Diffuse and specular reflections are added using the method of Phong[15] to calculate the highlights. On a 512 × 512 image with only about five or ten pixels per atom, this method is very cost effective. In addition, the calculations for regions in shadow are very cheap, which is fortunate since these shading calculations do, however, grow as the square of the resolution.

Regions in shadow are shaded by assuming that the sphere in question is part of a unit sphere. This means that all square roots involved in the calculation of the normals are less than 1. This in turn allows these calculations put be performed by table lookup instead of the costly square root function. Except for initial calculations, the entire piece of scan line is calculated using additions, a table lookup, and one multiplication (the z normal times the maximum intensity) by utilizing forward differences (see Max[10]).

The lit regions also use forward differencing but cannot be shaded as easily since the light source is no longer at the viewer's eye. For these calculations, performance is improved by using the square root lookup mentioned above. If the square is greater than 1, the standard square root routine is used. This compromise reduces the amount of time spent in the square root routine to an acceptable level.

SHADOWS USING TRAPEZOIDS: THE GENERAL CASE

Now let us try to remove our simplifying assumptions. The easiest to remove is the assumption that the sphere is centered at the origin. A generally placed sphere, centered at $c = (x_c, u_c, x_c)$ in the (x,y,v) coordinate system, has the equation.

$$(x-x_c)^2+(u-u_c)^2+ (v-v_c)^2=r^2$$

So if (x_0, u_1) is a point on the top or bottom of a trapezoid in the light source view, we can find

$$v_1 = (r^2-(x_0-x_c)^2-(u_2-u_c)^2)^{1/2}+v_c$$

and then proceed as before, rotating about a line through the center c, parallel to the x-axis.

Now consider the light source direction. If it does not lie in the YZ plane, an arc projecting onto a vertical scan line in the final view might not project to a straight line in the light source view. So we are unable to remove

this restriction. If other light source positions are required, the data can be rotated until the light source direction lies in the (x,y) plane and then the final raster image can be rotated back using the algorithm of Catmull and Smith.[16]

Now suppose neither the light source nor the viewer is at infinity, so that both viewers are perspective projections. An arc that projects to a vertical line in one view will again project to a curve in the other. But if the angle of view is small enough, this curve will be very close to a straight line. Even if the angle of view of the whole image is large, the angle subtended by each individual sphere is small. Thus, we still draw each sphere as if its profile projects to a circle instead of an ellipse, and use translation and scaling to make scan lines in the two views correspond.

INPUT

The input to the shadow program consist of lists of trapezoids for each atom, produced by the program ATOMLLL,[10] a version of the visible surface program of Knowlton and Cherry[9] from the view of the light source. To ease calculations, the 16-bit integers that are output by ATOMLLL are converted to floating-point numbers that map almost directly into screen space.

ATOMLLL generates perspective views so alignment of an atom in each view becomes a nontrivial task. (Removing the perspective was considered, but we felt that this would seriously degrade the quality of the final image.) First of all, since the light source is rotated about the x-axis, one would not expect the atoms in each view to have similar y centers. The addition of perspective also alters the x center and radius. Thus, the trapezoids in the light source view must be scaled and translated to bring them into alignment with those in the final image. All of these alterations must be performed on each trapezoid in the list before it can be used as a lighting mask for the front view. Because of the way in which trapezoids are described, this results in eight distinct cases, depending on which part of the trapezoid is being scaled.

Performing each of these calculations correctly for each situation is quite tedious, but is not mathematically challenging. Fortunately, the scaling and translation is only performed once. After the front view is read in, its (x,y) center and radius are passed along to the routine that reads the light source view. It compares these three values and performs any modifications as the trapezoid list is read in for processing.

OUTPUT

It was important to the application that this program be capable of rendering images on either a film recorder or an 8-bit frame buffer. This restriction prohibited many easy solutions to antialiasing and dictated a new intermediate format that could be used on either device.

Output is organized as vertical scan lines on an atom by atom basis, so that each visible atom is completely described before the next. Each scan line contains the x position, the y minimum and maximum, and a run of intensities. The intensity is an 8-bit value with the first 192 values representing pure color as reflected by a Lamberian surface. The remaining 64 values have an increasing amount of white added for highlights. This is similar to the simple shading model described by Rogers.[17]

For output to a 24-bit film recorder, eight lookup tables, one for each color of atom, were created. (Actually, there is room for up to 256 different colored atoms, but our needs never exceeded 5.) Each atom indexed into this table by its color number and intensity to yield its red, green and blue components for each pixel. Each component is written to a separate file so that the image can be recorded on film in three passes. This avoids frequent filter switching, which wastes time and strains the motor in the film recorder.

For output into an 8-bit frame buffer the lookup table was mapped into eight 32-color segments, one segment for each atom color. Twenty-four of the colors are used for pure color, and the remaining eight are used for highlights. The intensity recorded in the data file is divided by 8 so that it can be indexed into the lookup table using the atom color number as the high-order three bits. This solution yields color contour bands on large atoms, but these are not visible on a 512 x 512 window on a color Sun workstation. If necessary, a random dither could be used to remove the contour band, but since the 8-bit buffers were used mostly for previewing, no effort was made to correct this problem.

RESULTS

Depending on the complexity and the geometry of the molecule rendered, a single frame took anywhere from 2 to 20 seconds on a Cray XMP 4/16 for the hidden-surface calculation (independent of the resolution). To render on a VAX 785 or Sun 3/160 (the Sun being a little faster), A–DNA with 1275 atoms took 30 seconds for a 512 x 512 image and 20 minutes at 4096 x 4096. The primary cost was in the shading calculations. The Color Plates for this article were rendered at 2048 x 2048, well over the resolution of the 35mm film used and thus eliminating the need for antialiasing.

This software was used to create a movie, "Anchoring Unit of Protamine and DNA," which consisted of 412 atoms calculated over 1200 frames at 1024 x 1024 resolution. Each 300-frame segment took about 12 hours on a VAX 785 and included a great deal of tape I/O as frames were moved down from tape, rendered, and then moved back to tape in 50-frame segments. Each segment needed about 4 megabytes of trapezoid information describing the visible regions in each view and yielded about 14 megabytes of computed image. The 50-frame segmentation was chosen simply because this much disk space was usually available. It was more important to have the job run slower than to have it interrupted in the middle of the night because it ran out of space.

FUTURE WORK

A method for antialiasing that would improve both the frame buffer image and the film recorder image eluded us. The safest, cleanest, most efficient use of man and machine time for this endeavor turned out to simply be increasing the resolution of the computed image. A method for antialiasing based on fractional pixel coverage would allow lower resolution and less image data, making the producton of a film less burdensome.

The shading algorithm could also be highly parallelized. Since each atom need only know about its counterpart in the alternative view, we could, in theory, put one atom on each processor. The shading cost would then be the time to shade the worst atom, plus the time to write the image out to the frame buffer.

ACKNOWLEDGEMENTS

We are eternally indebted to Paul Renard for his help with the film recorder software. Without his help, the volume of data needed to make a movie on the film recorder would have increased tenfold. Rod Balhorn built the Protamine anchoring unit structure on UCSD's MIDAS system, written by Tom Ferrin and supported by Conrad Huang. Mike Zaklan ported UCSF's AMBER software onto our CRAY-2, which allowed us to perform the energy minimization and molecular dynamics calculations in our movie. John Blunden prepared and edited the film and advised about other aspects of preparing the raw footage. Thanks to Pat Gerling, our division secretary who got this released for publication in our absence from the LLNL. And special thanks to Rebecca Springmeyer, who was always willing to interrupt her own research to lend a hand, as good officemates always do.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract number W-7405-Eng-48, and was partially supported by an Institutional Research and Development grant.

REFERENCES

1 Duff, Tom. Smooth shaded renderings of polyhedral objects on raster displays. *Computer Graphics* 1979, **13**, 270–275

2 Pique, M.E. Fast 3D display of space-filling molecular models. Technical Report 83-004, Department of Computer Science, University of North Carolina, Chapel Hill, NC, 1983

3 Knowlton, Ken. Computer-aided definition, manipulation, and depiction of objects composed of spheres. *Computer Graphics* 1981, **15**(1), 48

4 Porter, Tom. Spherical shading. *Computer Graphics* 1978, **12**(3), 282–285

5 Max, Nelson. Computer representation of molecular surfaces. *IEEE Computer Graphics and Applications* 1983, **3**(5), 21–29

6 Fujimoto, Akira, and Takayuki, Tanaka. ARTS: accelerated ray-tracing system. *IEEE Computer Graphics and Applications* 1986, **6**(4), 16–26

7 Glassner, Andrew S. Space subdivision for fast ray tracing. *IEEE Computer Graphics and Applications* 1984, **4**(10), 15–22

8 Arvo, James, and Kirk, David. Fast ray tracing by ray classification. *Computer Graphics* 1987, **21**(4), 55–61

9 Knowlton, Ken, and Cherry, Lorinda. ATOMS — a 3D opaque molecule system — for color pictures of space-filling or ball-and-stick models. *Computers and Chemistry* 1977, **1**(3), 161–166

10 Max, Nelson. ATOMLLL: Atoms with shading and highlights. *Computer Graphics* 1979, **13**, 165–173

11 Williams, Lance. Casting curved shadows on curved surfaces. *Computer Graphics* 1978, **12**(3)

12 Reeves, Steve, Salesin, David H. and Cook, Robert L. Rendering anti-aliased shadows with depth maps. *Computer Graphics* 1987, **21**(4), 283–291

13 Weiler, Kevin, and Atherton, Peter. Hidden surface removal using polygon area sorting. *Computer Graphics* 1977, **11**(2) 214–222

14 Max, Nelson. Atoms with transparency and shadows. *Computer Vision, Graphics, and Image Processing* 1984, **27**, 46–63

15 Phong, Bui-Tuong. Illumination for computer generated images. Computer Science Department Report UTEC-CSs-73-129, NTIS ADA 008 786, Ph.D. Thesis, University of Utah (1973)

16 Catmull, Ed, and Smith, Alvy Ray. 3-D transformation of images in scanline order. *Computer Graphics* 1980, **14**(3), 279–285

17 Rogers, David F. *Procedural Elements For Computer Graphics*. McGraw-Hill, New York, 1985