

Nonparametric methods of calculating points on the curve produce the recently introduced superquadric objects at great savings in time.

Faster Calculation of Superquadric Shapes

Wm. Randolph Franklin and Alan H. Barr
Rensselaer Polytechnic Institute

Designers in CAD/CAM, modeling complex shapes on the computer, frequently create all their objects from a small library of fundamental objects whose properties are well understood. Thus, PADL,^{1,2} a solid-object design system, uses rectangular boxes and cylinders. Another system, EUKLID,³ uses prisms, anti-prisms, boxes, and so on. In the first issue of *IEEE Computer Graphics and Applications*, Barr⁴ introduced a new family of parametric objects called superquadric shapes. They are generalizations of conic sections and toruses with the exponent, formerly 2, replaced by an arbitrary positive number (see Figures 1 and 2). Thus, they are 3-D versions of Piet Hien's superellipses. These curves are also mentioned by Faux and Pratt⁵ and by Flanagan and Hefner.⁶ Figure 3 shows superellipses with nine different exponents from

0.1 to 10. These objects are being used in solid modeling and display on color raster graphics equipment because they are flexible enough to represent a broad family of shapes, and because they are easy to calculate. The pictures in Barr's article show the wide variety of useful shapes that can be created with generalizations of toroids and hyperboloids of one and two sheets. The shapes can be rounded, squared, or pinched, and can have different properties in different directions.

Superquadrics can be defined by either implicit or parametric equations. The parametric form is used for calculation, since surface points and normal vectors can be generated more easily with this method than with implicit equations, but the sampled points are sometimes very unevenly spaced. We have used computational ge-

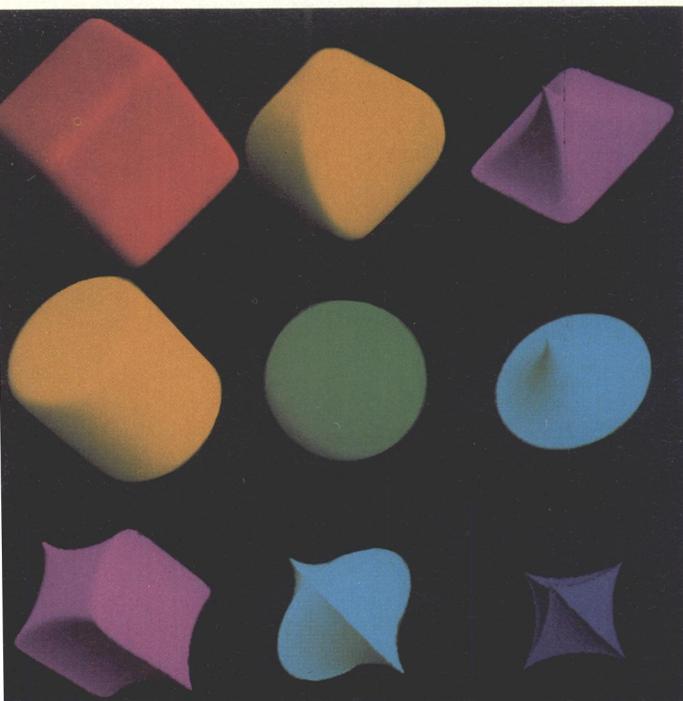


Figure 1. Superquadric ellipsoids.



Figure 2. Superquadric toroids.

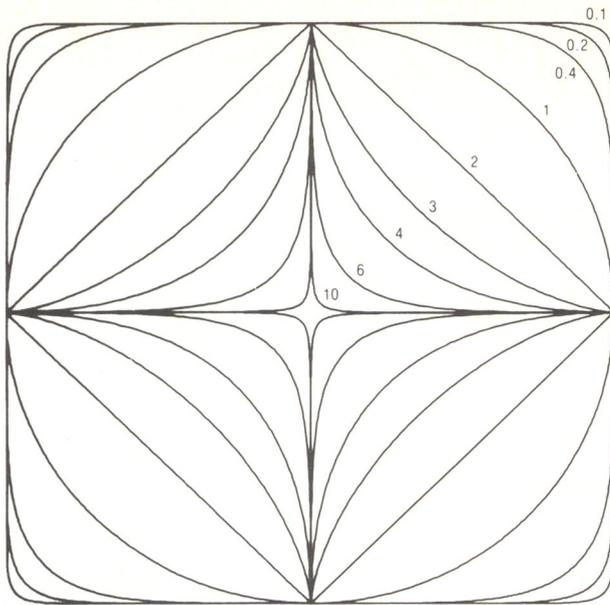


Figure 3. Superellipses: $x^{2/\epsilon} + y^{2/\epsilon} = 1$ for $\epsilon = 10, 6, 4, 3, 2, 1, 0.4, 0.2,$ and 0.1 .

ometry techniques^{7,8,9,10,11} to develop more efficient methods for generating points on the curve with explicit equations or series approximations. These methods have three advantages:

- They naturally produce equally spaced points.
- They require much less CPU time to produce each point.
- They can produce surfaces of greater generality.

Now, why do we want equally spaced points? The superquadrics have a smaller radius of curvature at the corners, where the parametric method generates more closely spaced points—that is desirable.

The answer is that the parametric method does not generate points whose spacing depends on the local radius of curvature; the variation is much more extreme. Figures 4(a) to 4(i) show equal variations of the parameter, θ , for different values of the exponent, ϵ ; θ varies in increments of 10° , so that the superellipse is approximated by 36 points. The following table shows how unevenly arc

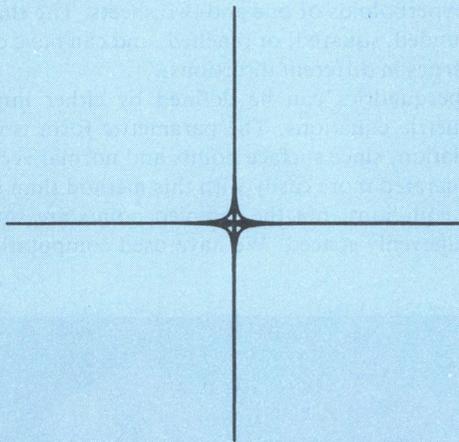


Figure 4(a). Superellipse: $\epsilon = 10$, varying θ in 10° increments.

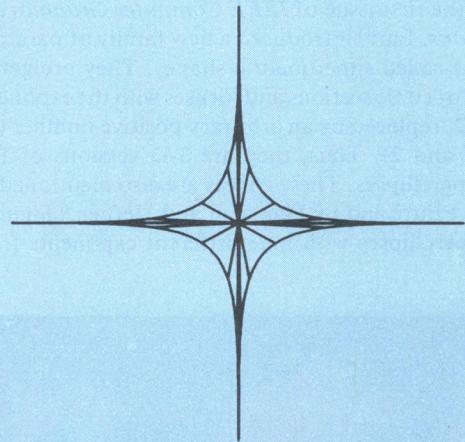


Figure 4(b). Superellipse: $\epsilon = 6$, varying θ in 10° increments.

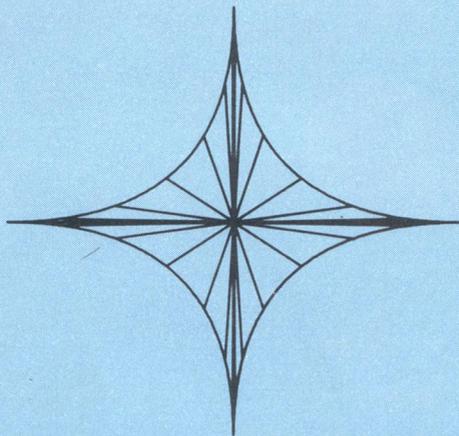


Figure 4(c). Superellipse: $\epsilon = 4$, varying θ in 10° increments.

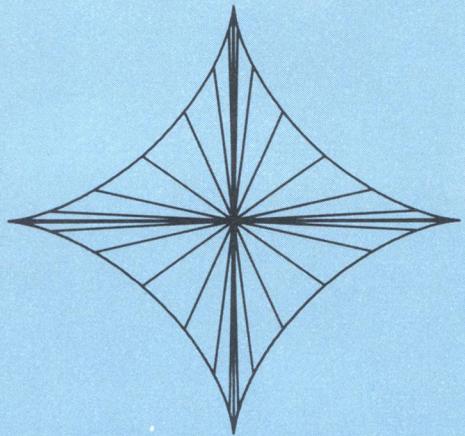


Figure 4(d). Superellipse: $\epsilon = 3$, varying θ in 10° increments.

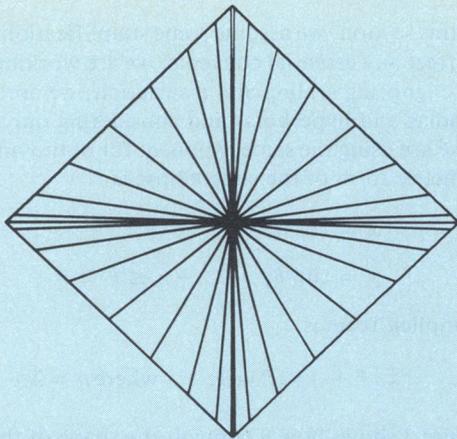


Figure 4(e). Superellipse: $\epsilon = 2$, varying θ in 10° increments.

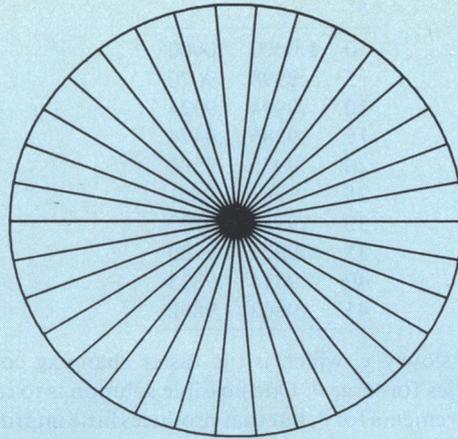


Figure 4(f). Superellipse: $\epsilon = 1$, varying θ in 10° increments.

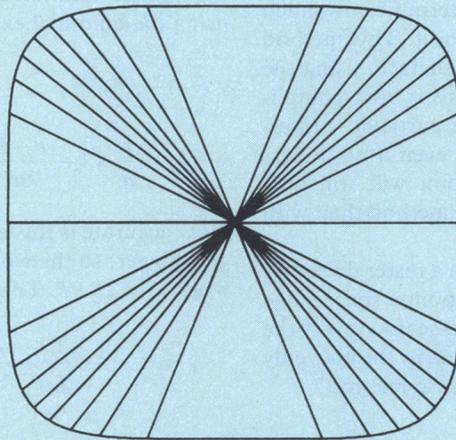


Figure 4(g). Superellipse: $\epsilon = 0.4$, varying θ in 10° increments.

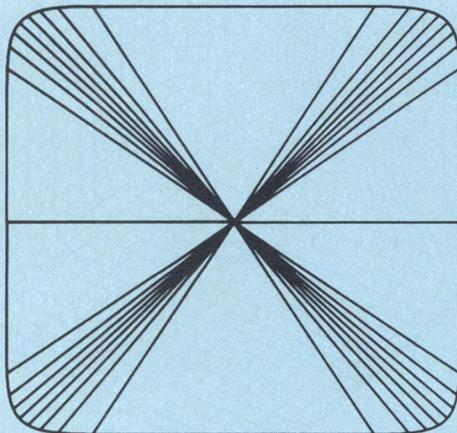


Figure 4(h). Superellipse: $\epsilon = 0.2$, varying θ in 10° increments.

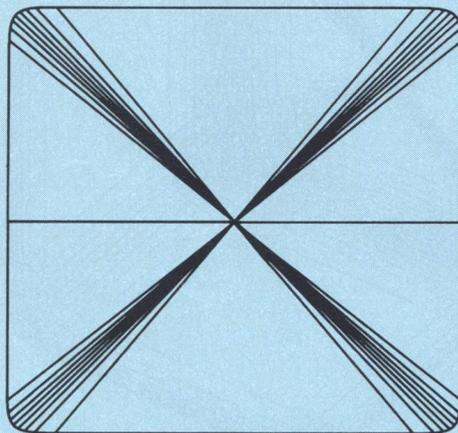


Figure 4(i). Superellipse: $\epsilon = 0.1$, varying θ in 10° increments.

length varies with θ for $x = \cos^{.02} \theta$, $y = \sin^{.02} \theta$:

θ°	x	y
0	1.0000	.0000
5	.9998	.9070
10	.9994	.9324
15	.9986	.9474
20	.9975	.9580
25	.9961	.9661
30	.9943	.9727
35	.9921	.9780
40	.9894	.9825
45	.9862	.9862

Note how slowly y , which is the faster changing coordinate, varies for large θ . One possible solution is to take smaller increments for θ , but that produces little improvement in covering the large gap. Thus, generating smooth curves with the raw parametric method requires calculating many points, especially when using exponents that make very square shapes.

Now, it is possible to calculate the superquadric's local curvature and the rate of change of arc length as the parameter varies, and from there to determine how much to vary the parameter for each new point. This method produces an optimal sampling of the surface, using the fewest number of points. However, this is both complicated and strongly dependent on the particular superquadric being calculated. In any case, even if no useless points are calculated, each useful point will still take longer to calculate with the parametric method than with the explicit method.

Finally, plotting the superquadric on a raster device by the explicit method, we can produce points spaced one pixel apart, and then we don't have to scan-convert the piecewise linear approximation generated by the unevenly spaced points produced by the parametric method.

The explicit equation for a superellipse

In this section, we assume some simplifications that do not affect any essential concepts: we are working in 2-D; we are ignoring scaling and translation; we are ignoring parabolas and hyperbolas and considering only ellipses; and we are using the same exponent for both x and y . The parametric form of the superellipse is

$$\begin{aligned} x &= \cos^\epsilon \theta \\ y &= \sin^\epsilon \theta, \quad -\pi \leq \theta < \pi \end{aligned}$$

The implicit form is

$$|x|^n + |y|^n = 1, \quad \text{where } n = 2/\epsilon$$

Figure 3 shows how n controls the shape of the curve: $n = 2$ gives a circle, and as $n \rightarrow \infty$, the curve approaches a square; $n = 1$ gives a diagonal square, and as $n \rightarrow 0$, the curve becomes more and more pinched. Since the curve has eightfold symmetry, let us calculate only the piece with $0 \leq x < y$ and then reflect it seven times. We obtain the explicit equation by expressing y as a function of x and using the binomial expansion:

$$y = (1 - x^n)^{1/n} \tag{1}$$

$$= 1 - \frac{1}{n} x^n - \frac{n-1}{2n^2} x^{2n} - \frac{(n-1)(2n-1)}{6n^3} x^{3n} - \dots$$

How accurate is this expression? It gets less accurate as x gets bigger, so the worst case occurs when $x = y$, that is, when $x = 2^{-1/n}$. Then,

$$y = 1 - \frac{1}{2n} - \frac{n-1}{8n^2} - \frac{(n-1)(2n-1)}{48n^3} - \dots$$

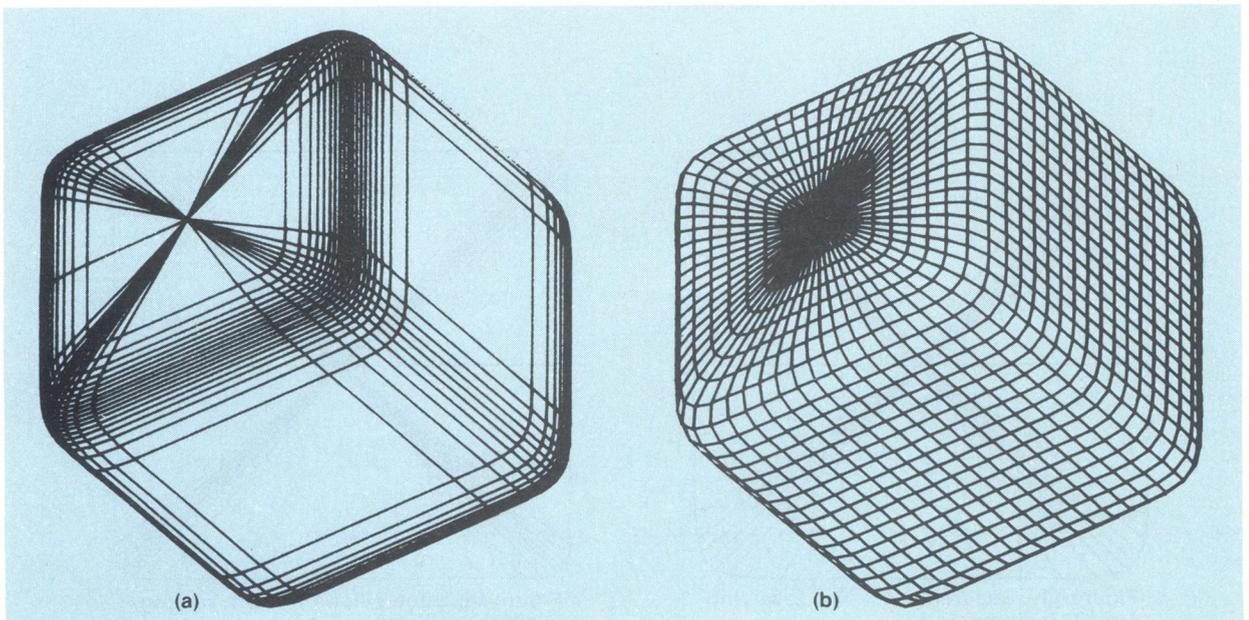


Figure 5. Superquadric cuboid: (a) raw parametric sampling of surface and (b) explicit sampling of surface.

Thus the explicit equation becomes more accurate as n increases, that is, as the parametric method becomes less suited. Consider, for example, $n = 5$, i.e., $\epsilon = 0.4$, which gives a mildly rounded superellipse, as shown in Figure 4(g). The terms of the above equation, for the worst point, are

$$y = 1 - .1 - .02 - .006 - .0021 - .000798 - \dots$$

Thus after only five terms, the partial sum is accurate enough for display on a 1000-by-1000 screen. If we want a squarer curve, which means a higher n , even fewer terms will suffice. For most points with a given n , such as $n = 5$ and $x = 0.5$, the series converges much faster than above:

$$y = 1 - .00625 - .0000781 - \dots$$

Thus a mere two terms will suffice on a 1000-by-1000 screen to calculate this point. The n can range up to 346 before the ellipse becomes a square at our resolution, and there the series converges in one or two terms for all x .

For small n , the binomial series converges too slowly, so it is faster to use the exact equation (1). The exact equation is still better than the raw parametric method, since it requires only two exponentiations, while the parametric method requires two exponentiations and two trigonometric evaluations. The exact equation also produces equally spaced points. We could also use the binomial approximation for small x on a given curve, and then switch to the exponentiation functions when x is larger and the series converges more slowly.

It is useful to allow the exponents of x and y to differ. The only effect of this is to reduce the degree of symmetry from eightfold to fourfold.

In summary, the explicit equation is faster than the parametric method for calculating superellipses for all values of n .

The explicit equation for a superquadric

Calculating a superquadric in 3-D is similar to calculating a superellipse in 2-D. The parametric form is

$$x = \cos^{\epsilon_1} \eta \quad \cos^{\epsilon_2} \omega \quad -\pi/2 \leq \eta \leq \pi/2$$

$$y = \cos^{\epsilon_1} \eta \quad \sin^{\epsilon_2} \omega \quad -\pi \leq \omega < \pi$$

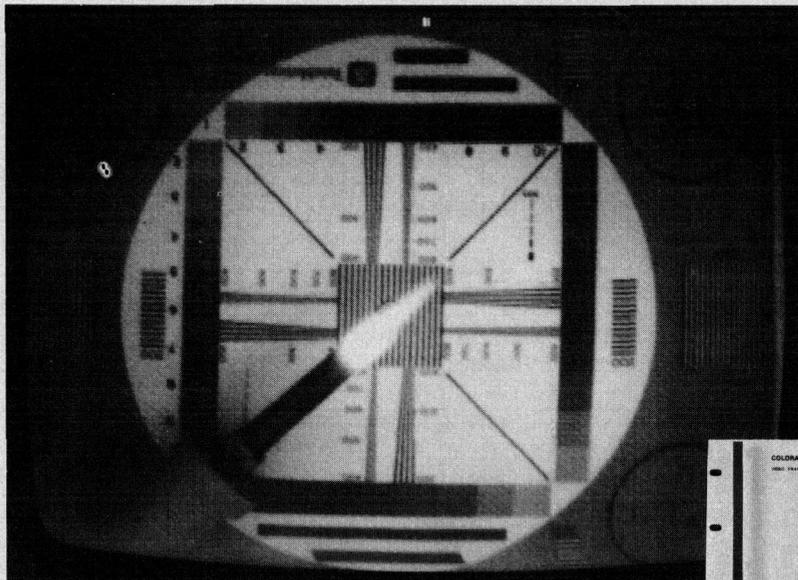
$$z = \sin^{\epsilon_1} \eta$$

The surface is generated at each value of η and ω from two superelliptic curves—one associated with η , and one associated with ω (see Figure 5a). The speediest way to generate the surface points is to precompute and store the superellipses by the explicit method, because only two multiplies and three retrievals are needed for each superquadric surface point (see Figure 5b). This property is true for any surface that can be represented in the form of a spherical product. It is also true for the superquadric normal vectors. For cases in which $\epsilon_1 = \epsilon_2$, the explicit equation becomes

$$z = (1 - x^n - y^n)^{1/n}$$

We will evaluate it for $0 \leq x \leq y \leq z \leq 1$, which is 1/48 of the superquadric, and reflect it 47 times. If $n \gg 1$, then we can expand the binomial series as before. Note that the

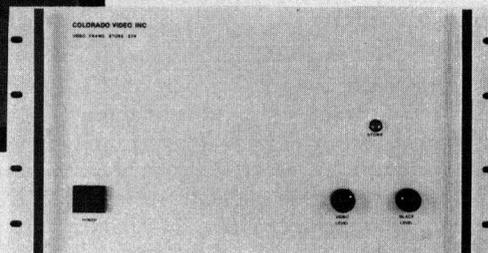
Digital Video Memory



Seeing by the light of a soldering iron, one of the special applications of the 274C.

The 274C contains a 512 x 512 x 8-bit memory that may be used to "freeze" black-and-white or color TV signals. A digital I/O port allows rapid transfer of memory contents to a computer or other location, as well as allowing use of the 274C for high quality reconstruction of digitally encoded images.

Options include multiple memory configurations and image subtraction. Please contact us for price and delivery information.



colorado video

POST OFFICE BOX 928 BOULDER, COLORADO 80306 USA
PHONE (303) 444-3972 TWX 910-940-3248 (COLO VIDEO BDR)

parametric form obscures the 48-fold symmetry of this type of superquadric because spherical coordinate systems have a distinguished orientation. Using the exact explicit equation requires three exponentiations per point, compared to the raw parametric method's four trigonometric evaluations plus four exponentiations—167 percent more function calls. However, the spherical product parametric representation requires only two multiplies per surface point. As before, the binomial series is even faster when it converges efficiently. And, as before, we can generate points at exactly equal intervals in x or y , a requirement which the spherical product form is not well-suited to meet.

If we relax the restriction that the exponents be equal, then the implicit equation can be generalized to

$$x^{n_1} + y^{n_2} + z^{n_3} = 1$$

This equation is more general than the parametric equation, since that has only two degrees of freedom, ϵ_1 and ϵ_2 , which forces $n_2 = n_3$. However, there is no spherical product that corresponds to this form, and the normal vectors do not have the same functional form as the position vectors, although they are easy to calculate from the $(n-1)$ th powers of x , y , and z . Letting the exponents differ reduces the degree of symmetry, so we must calculate more points. That is also the major effect of calculating hyperboloids, paraboloids, and toroids, as well as ellipsoids.

We frequently wish to generate surface points or voxels of a superquadric so that a solid modeling system can then manipulate the object. A sphere filling a 1000-by-1000-by-1000 grid has about 2,000,000 surface points, and inserting so many points into a data base may be expensive. If we wish only to plot a set of the superquadrics and not intersect them, then it is much more efficient merely to write their points into the Z-buffer and never to create the data base at all, saving the large amount of space that such a data base would require.

Generally oriented superquadrics

The preceding derivation assumed an orthogonal cubical superquadric. The easiest way to add scale factors and arbitrary orientation is to generate a point (x,y,z) of the standardized superquadric for the desired parameter n , and then generate and transform the reflections. We can then postprocess them as desired, for example sorting them back-to-front for writing into a Z-buffer. Since the program uses a lot of virtual memory, we must process the data in order as much as possible, to reduce the size of the program's working set and minimize page faulting.

For generating a sequence of related superquadrics with the same parameter, n , as in an animated sequence showing an assembly of parts opening up, there are two ways to speed up the process. If it is faster to read a page of points from disk than to calculate them, we might calculate once and store a superquadric in some standard orientation, and then just reread and transform its points. Or, after we calculate a point, we might calculate all its transformations and, after all the points and their trans-

formations have been calculated, sort and bring together all points for each transformation.

Using lookup tables

Evaluation of superquadric surface points and normal vectors is slow because they require so many exponentiations. One way to increase speed would be to attach to the bus a peripheral processor that does nothing but exponentiations. For images, an easier method is to use lookup tables, since we only need to resolve to 0.001 accuracy. There are two ways we might use lookup tables other than the previously mentioned spherical product method. Suppose that we must plot

$$x^5 + y^5 + z^5 = 1$$

We can calculate a table of $\beta = \alpha^5$, tabulated from 0 to 1 in steps of 0.001, and another table of $\alpha = \beta^{0.2}$. Then calculating each $z(x,y)$ takes three lookups, two subtractions, and no floating-point operations or function calls. Precalculating and storing the table takes insignificant time and space compared to the superquadric itself.

The other method is to precalculate tables of exponentials and logs. Then we can evaluate α^β by calculating $e^{\beta \cdot \ln(\alpha)}$. This method is slower because of the multiplication, and it is possibly less accurate because we must interpolate into a more rapidly changing function table for the second lookup. However, one set of tables works for all powers.

If we proportionally increase the number of points that we calculate as we increase the accuracy, then no matter what the accuracy, the lookup tables will always take an insignificant fraction of time and space. However, if we wish to calculate only a few points to a high accuracy, possibly because we wish to postprocess the superquadric without compounding roundoff errors, then the lookup tables might become too large a fraction of the total data structure. Then we would revert to the explicit equation or series approximation.

Implementation results

We implemented, on a Prime 750, six different methods of calculating the points on superquadrics:

- (1) the raw parametric method without using spherical products;
- (2) the exact explicit equation;
- (3) the series approximation, using three terms;
- (4) the above series approximation, using a lookup table for $\lg(\alpha)$;
- (5) the parametric method, using a spherical product with lookup tables for $\cos^{\epsilon_1} \eta$, etc.;
- (6) the exact explicit equation, using lookup tables for α^n and $\alpha^{1/n}$.

The execution time to calculate the coordinates of a point was isolated and measured. In each case, the calculation was repeated 10,000 times. Even so, the times varied about 10 percent from run to run. For the table lookups, the tables were assumed to be precalculated,

since in practice the time to calculate them is completely dominated by the rest of the program. The calculation times and their ratios to the time of the raw parametric method are as follows:

Method	CPU time (sec.) for 10,000 points	# times as fast as method 1
1	8.61	1.00
2	4.46	1.93
3	3.20	2.69
4	1.69	5.09
5	0.158	54.10
6	0.039	219.00

Thus, the time to calculate the points on the superquadric is reduced from a dominant problem into insignificance.

Techniques from traditional computer science, such as numerical analysis and data structures, can be brought to bear on computationally difficult problems in computer graphics to great effect. In the case of superquadrics, by changing from a parametric method to an explicit equation, we can produce evenly spaced points, faster, for a surface of greater generality. Thus, solid modelers can use flexible families of shapes at much lower cost than before. ■

References

1. R. B. Tilove, "Set Membership Classification: A Unified Approach to Geometric Intersection Problems," *IEEE Trans. Computers*, Vol. C-29, No. 10, Oct. 1980, pp. 874-883.
2. H. Voelcker, et al., "The PADL-1.0/2 System for Defining and Displaying Solid Objects," *Computer Graphics*, Vol. 12, No. 3, Aug. 1978, pp. 257-263.
3. M. Engeli and V. Hrdlicza, *EUKLID Eine Einführung*, FIDES Rechenzentrum, Treuhand Vereinigung, Bleicherweg 33, 8002 Zürich, 01/25 7840, Feb. 1974.
4. A. H. Barr, "Superquadrics and Angle-Preserving Transformations," *IEEE Computer Graphics and Applications*, Vol. 1, No. 1, Jan. 1981, pp. 11-23.
5. I. D. Faux and M. J. Pratt, *Computational Geometry for Design and Manufacture*, Wiley, New York, 1979, p. 230.
6. D. L. Flanagan and O. V. Hefner, "Surface Moulding—New Tool for the Engineer," *Aeronautics and Astronautics*, Apr. 1967, pp. 58-62.
7. W. R. Franklin, "Evaluation of Algorithms to Display Vector Plots on Raster Devices," *Computer Graphics and Image Processing*, Vol. 11, No. 4, Dec. 1979, pp. 377-397.
8. W. R. Franklin, "A Linear Time Exact Hidden Surface Algorithm," *Computer Graphics (ACM Siggraph '80 Conf. Proc.)*, Vol. 14, No. 3, July 1980, pp. 117-123.
9. W. R. Franklin, "An Exact Hidden Sphere Algorithm That Operates in Linear Time," *Computer Graphics and Image Processing*, Vol. 15, No. 4, Apr. 1981, pp. 364-379.
10. W. R. Franklin, *Efficient Polyhedron Intersection and Union*, Rensselaer Polytechnic Institute, Image Processing Lab, IPL-TR-81-001, Jan. 1981.
11. W. R. Franklin, *Efficiently Computing the Haloed Line Effect for Hidden Line Elimination*, Rensselaer Polytechnic Institute, Image Processing Lab, IPL-TR-81-004, Dec. 1980.

Acknowledgments

This article is based upon work supported by the National Science Foundation under grants ENG79-08139 and ISP79-20240.

We would like to thank Jeff Goldsmith and Bruce Edwards for technical assistance with the illustrations.



Wm. Randolph Franklin is an assistant professor with the Electrical, Computer, and Systems Engineering Department at Rensselaer Polytechnic Institute. His research interests include computational geometry techniques for the design of efficient algorithms to manipulate large graphic data bases with high precision in operations such as hidden surface removal.

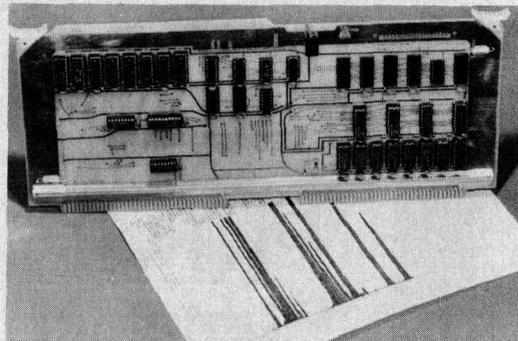
Franklin received the BSc (honors) in computer science from the University of Toronto, Canada, and the AM and PhD in applied mathematics from Harvard University. He is a member of the IEEE, the ACM, and the L-5 Society.



Alan H. Barr, a doctoral student in applied mathematics at Rensselaer Polytechnic Institute, is a member of the research staff at the institute's Center for Interactive Computer Graphics. He is investigating the mathematical modeling and simulation of mechanical and biological processes, using computer animation to communicate time-dependent and three-dimensional results.

Barr received the BS and the MS in mathematics from RPI. He is a student member of SIAM, AAAS, and the American Society for Cell Biology.

OMNI-GRAPHICS Transforms your TI Model 810 into a high-speed printer/plotter



- raster graphics and all 810 functions
- 2 plotting modes with 1584 x 792 dots per page
- programmable expansion in print and plot
- you may load your own 75 character software font and recall it for print by standard ASCII
- all standard ASCII characters 150 characters/sec
- bidirectional printing and plotting
- rates 110 to 9600 baud serial RS232 or parallel interface
- no electrical or mechanical mods
- TI warranties are preserved
- standard paper and built-in self test

OMNI-GRAPHICS and the 810 provide low cost, high speed plotting.

ANALOG TECHNOLOGY CORPORATION
15859 E. Edna Place
Irwindale, CA 91706 (213) 960-4004