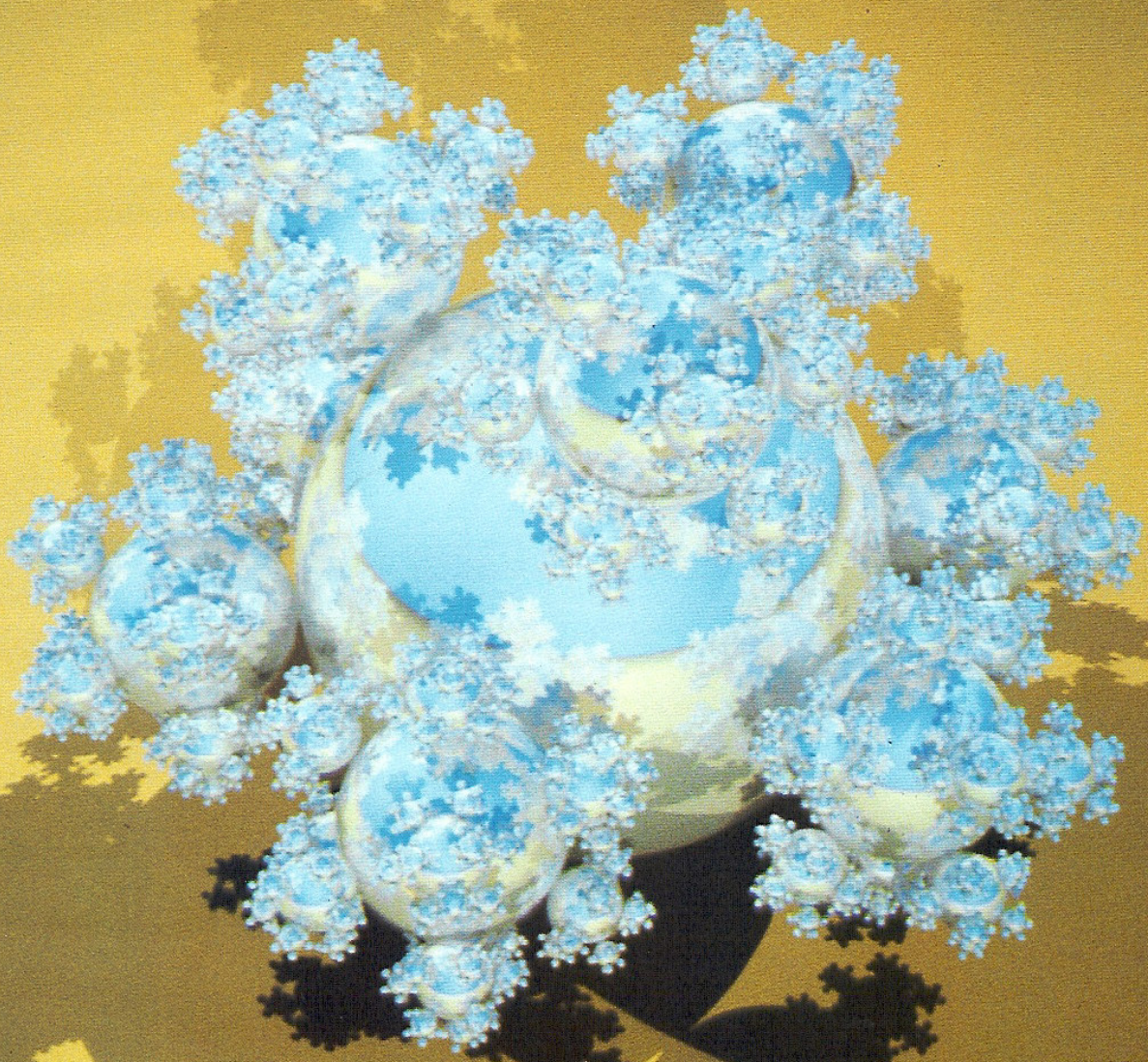
IEEE

Computer Graphics and applications

NOVEMBER 1987



Smooth Surfaces
Surface-Rendering Algorithm
Image Composition
Sweeping Technique
CAD with IGES

THE COMPUTER SOCIETY OF THE IEEE
THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Displays on Display

Editor: Frank Crow

A Proposal for Standard Graphics Environments

Eric Haines, 3D/Eye Inc.

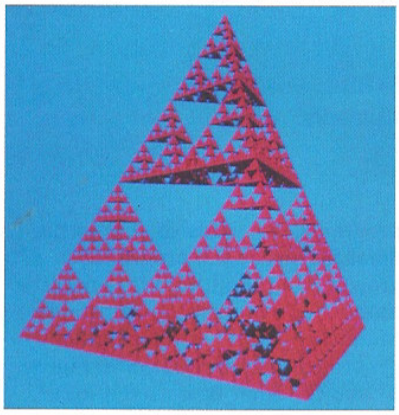


Figure 1. Recursive tetrahedral pyramid.

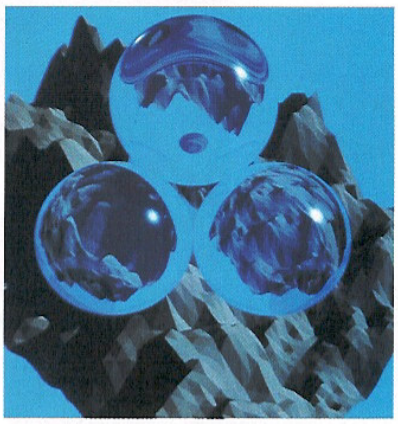


Figure 2. Fractal mountain with spheres.

One concern of the computer graphics community has been the efficiency of rendering algorithms. Buyers faced with a variety of hardware graphics accelerators would like to know the average display rate of each machine. In such fields as ray tracing, researchers continue to explore which is the fastest way to find the closest intersection point for a ray and a set of primitives. The problem faced by these and other people involved in computer graphics is a lack of standards.

Within the hardware field there has been some progress in solving this problem. A frequently quoted number for a graphics accelerator is the number of polygons per second that can be output to the screen. A number of companies use the same standard, defining an average polygon to be a randomly oriented square which is 10×10 pixels in size. This is a useful metric of one kind of raw processing power. Some other measurements have also been used, such as the relation between polygon size and the polygon draw rate.

In such research areas as ray tracing, a second approach is used. The rendering of a single primitive, such as a polygon, can be affected by the other primitives in the environment. For example, another primitive could cast a shadow or be seen in reflection from the primitive being rendered. This has led to timing comparisons based on the time for calculating ray intersections, instead of a primitives-per-second rate. One of the problems with trying to compare ray intersection times is that there are almost no standard test environments. One researcher will ray trace a car; another, a tree. The question arises, "How many trees to the Camaro?"

My proposal is that we should all be using the same environments. I originally heard of this idea from Don Greenberg while I was in Cornell's Program of Computer Graphics. He and Ed Catmull had once discussed producing some environments that would be used as standards for testing rendering algorithms. A few years later, Tim Kay presented a paper on efficient ray tracing at SIGGRAPH 86. He offered his database descriptions to any researcher who wanted to use them. Discussions with him and other researchers led me to create a number of scenes for testing ray-tracing algorithms.

The databases are fairly familiar and "standard" to the graphics community. The scenes are generated by the "Standard Procedural Database" package, or SPD for short. Each database is generated by a program written in C. The output of the program is in text, with information about the view, lighting conditions, and primitives being output in a simple format. Presently polygons, polygonal patches (polygons with a different surface normal at each vertex), spheres, cylinders, and cones are supported. The researcher has to write a program to translate these simple output data into the format needed by the algorithm or hardware being tested.

The programs use simple rules to create complex databases. One advantage of using programs to

0272-1716/87/1100-0003$01.00 © 1987 IEEE

Figure 3. Tree.

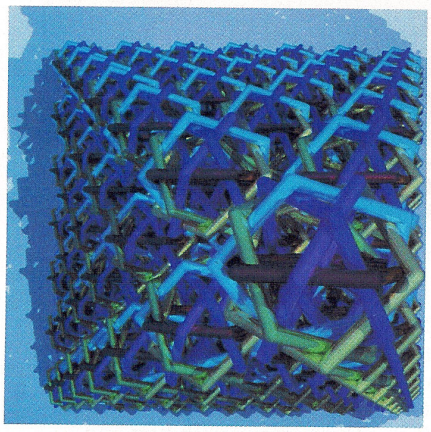


Figure 4. Dodecahedral rings.

generate databases is that the programs can be fairly short. For example, the average SPD generator program is about three pages of code, not including the common library of routines. Another advantage is that the geometric description of the output databases can be controlled directly. One example is that curved surfaces can also be output as sets of polygonal patches. This allows the database to be used to test both ray-tracing algorithms, which normally would use the simple geometric description, and hardware graphics accelerators, which tend to use polygonalized representations.

If you represent the databases by programs, you can also change the database size. Each program has a growth factor built into it so that the number of primitives is adjustable. Mandelbrot and Norton originally generated the tetrahedron in Figure 1,[1] and Glassner first used it for testing ray tracing.[2] Kay and Kajiya also used it,[3] as did Arvo and Kirk.[4] There are some 4096 triangles in the scene. Various researchers have used different numbers of tetrahedrons filling the same sized space. By changing the growth factor in the generating program, you can change these variants. Another advantage of the growth factor is that you can examine the relationship between an algorithm's performance and database.

The fractal mountain in Figure 2 is another common scene, inspired by Peter Watterberg's work on the cover of the advance program for SIGGRAPH 85. The glass balls are in the environment so that algorithms would have to deal with refraction and reflection rays. To test hardware graphics accelerators, the spheres are polygonalized.

Aono and Kunii's method is used in Figure 3 to generate a tree,[5] consisting of a total of 9190 spheres and cones. Each database in the SPD package has a default database size of close to 10,000 primitives, which was chosen as representative of an average-sized scene. Since each database program has a growth factor, the database size can be increased to provide more complicated environments for future testing.

The rings database in Figure 4 is one of the lengthiest test scenes to render. The scenes in the SPD package were chosen to generate very different kinds of space, with different surface properties, different numbers of lights, and different background characteristics. The rings scene takes a long time to ray trace because of some of these factors. It is highly reflective, all rays from the eye hit some primitive, and a large number of shadow rays are generated.

Figure 5 is a set of gears. This scene was designed to test how various algorithms and kinds of hardware deal with large, many-sided, concave polygons. It also includes a large number of reflection and refraction rays.

I call the object in Figure 6 a "sphereflake," since it is something like a three-dimensional snowflake curve. This scene was ray traced by the AT&T Pixel Machine in about half a minute—a rather amazing feat! This timing brings up one difficulty of comparing researchers' results. Comparison between different hidden-surface graphics accelerators can be made in a straightforward manner using these databases. Given the conditions for performing timing tests of the databases (which are included as a part of the SPD package), a hardware accelerator will perform at a certain absolute rate in terms of polygons per second. An advantage of the SPD package over the randomly oriented polygon test is that such hardware features as lights, specular highlighting, and transparency are also tested.

Ray-tracing algorithms should not be compared using raw speed, since the software is run on different machines and in different languages. Adjustments for these factors can theoretically be factored into the speed (e.g., LINPACKS testing), but megaflops ratings do not necessarily reflect true machine performance. Because of this, researchers often record such additional factors

Figure 5. Meshed gears.

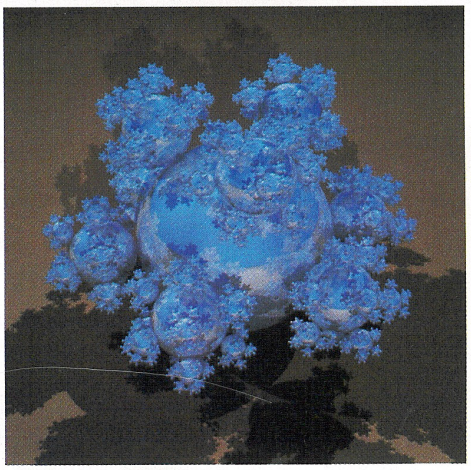


Figure 6. Sphereflake.

as the number of ray/object intersection calculations performed. When discussed in these terms, two different ray tracers can be compared on a more meaningful basis. By understanding the performance of different ray-tracing algorithms for a variety of scenes, the factors that affect performance of the algorithms can be recognized. For this reason, the SPD package is meant to be taken as a whole, with each scene having its own distinct characteristics.

The SPD package is in the public domain and can be accessed in a number of ways. Netlib[6] is distributing the package for free. For those with access to the Arpanet, write to "netlib@anl-mcs.arpa." If electronic mail on the UNIX UUCP network is available, write to "research!netlib." In either case, send the one-line message, "Send Haines from graphics."

An early, incomplete version of the SPD package was printed as an appendix in the notes for the "Introduction to Ray Tracing" course given at SIGGRAPH 87. For the IBM PC, the package may be available on a 360K 5-1/4" floppy disk. Write to me for details. If none of these media are available to you, send $4 for the latest printed version from me: Eric Haines, 3D/Eye Inc., 410 E. Upland Rd, Ithaca, NY 14850.

Presently this package is simply a proposal. Your feedback is needed on a number of questions, such as what constitutes an average scene for your applications, what primitives do you use, and what are your opinions on the package in general? Timings and statistics for graphics accelerators and different ray-tracing algorithms are also most welcome. My electronic mail address: hpfcla!hpfcrs!eye!erich@hplabs.HP.COM. ■

References

1. B.B. Mandelbrot, *The Fractal Geometry of Nature*, W.H. Freeman, New York, 1983, p. 143.
2. A.S. Glassner, "Subdivision for Fast Ray Tracing," *CG&A*, Oct. 1984, pp. 15-22.
3. T.L. Kay and J.T. Kajiya, "Ray Tracing Complex Scenes," *Computer Graphics* (Proc. SIGGRAPH 86), Aug., pp. 269-278.
4. J. Arvo and D. Kirk, "Fast Ray Tracing by Ray Classification," *Computer Graphics* (Proc. SIGGRAPH 87), July, pp. 55-64.
5. M. Aono and T.L. Kunii, "Botanical Tree Image Generation," *CG&A*, May 1984, pp. 10-34.
6. J.J. Dongarra and E. Grosse, "Distribution of Mathematical Software via Electronic Mail," *Comm. ACM*, May 1987, pp. 403-407.

Eric Haines is a senior software engineer at 3D/Eye in Ithaca, New York. His interests and responsibilities include creating a ray-tracing system for image synthesis. Previously he was a software engineer at RCA Astro-Electronics.

Haines received a BS in computer science from Rensselaer Polytechnic Institute in 1980 and an MS in computer graphics from Cornell University in 1986. He is a member of ACM.

He can be contacted at 3D/Eye, 410 E. Upland Rd., Ithaca, NY 14850 or hpfcla!hpfcrs!eye!erich@hplabs.HP.COM.