MATTHEW R. FEINSTEIN

THREE-DIMENSIONAL RENDERING AS A TOOL IN SCIENCE AND ENGINEERING

Specialized graphics hardware and standardized software libraries are making three-dimensional rendering a rapid and easy process. These advances have opened the way to using rendering to perform computations in short-wavelength scattering and propagation.

INTRODUCTION

As the power and capabilities of specialized computer graphics hardware have increased, computer-generated three-dimensional (3D) renderings of objects and scenes have become remarkably realistic, acquiring complex textures, perspective, lighting, atmospheric effects, and other characteristics that can all be varied in real time. From the standpoint of a scientist or engineer, however, an imbalance exists between graphics capabilities and graphics uses: we can render 3D scenes and objects rapidly and realistically, but the considerable programming effort and computer power that go into producing these effects contrast unfavorably with their limited (although commercially valuable) uses.

Recent trends nevertheless suggest that the computations underlying the entertaining graphics we see in movies and video arcades will become useful in science and engineering. Although specialized 3D graphics software and hardware have been available to graphics professionals for several years, the advances in technology that are lowering the cost, increasing the speed, and broadening the availability of personal computers are also putting sophisticated graphics tools on the desktops of many scientists and engineers.

These tools have a broader utility than one might expect, since calculations repeatedly and quickly performed in modern 3D rendering are very similar to computations done (or that would be done if they were not so arduous) in many technical problems. For example, a classic difficult problem in short-wavelength scattering is the effects of shadows and blockage of one object by another, but computations of shadows, hidden surface removal, and viewable areas are easy and rapid with modern graphics workstations. Similarly, the effects of nonuniform lighting, illumination and shadows on curved surfaces, and averaging of multiple viewpoints of a scene involve difficult, time-consuming computations that are part of the repertoire of advanced computer rendering techniques.

In the past, using dedicated computer graphics hardware for scientific computations would have required specialized knowledge. Major improvements, such as the development of hardware- and vendor-independent libraries and the emergence of standards for 3D rendering, have simplified the software interface between the programmer and the hardware. Trends in technology have brought high-powered graphics programming to nonspecialists, allowing the use of graphics hardware and software for many scientific and engineering purposes.

A CASE STUDY

The broader availability and lower cost of graphics tools lead one to ask when one should use advanced graphics hardware and software for computations. The trade-offs in performance, cost, and ease of operation are illustrated by a recent case study.

A large, mature Fortran program was being used to render 3D scenes for a variety of analysis tasks. However, the analyst using the program was unhappy with its performance. Since the Fortran program was written for a generic UNIX computer but was running on a Silicon Graphics (SGI) workstation, it was reasonable to propose that the rendering task be done with the built-in SGI graphics hardware and graphics software libraries. This approach raised several questions. For example, what is the difference in performance between a generic program and one that uses specialized hardware and software? What is the difference in performance between low-end and high-end graphics workstations?

We answered these questions by writing a functional equivalent of the older Fortran program using the same input files. The new program produced the same output files as the Fortran program but took advantage of the sophisticated graphics software library known as the GL, which comes with SGI computers. With the GL version in hand, the old and new programs could be compared on various SGI computers at APL.

Table 1 lists the times taken to produce a scene with about 1 million facets for the two programs tested on two SGI computers. One can see that rendering speed depends strongly on computer cost, and, in particular, that a significant performance advantage exists for the GL program, but it appears only at the high end of the cost spectrum. One notable finding not displayed in the table is that the GL program was at least an order

Table 1. Comparison of rendering speeds using two programs and two computers for a 1-million facet scene.

Computer	Generic program time (s)	GL program time (s)
SGI 4D/35 (low cost)	105	110
SGI Onyx (high cost)	30	5

of magnitude smaller and simpler than the older Fortran program.

We have learned that rendering is a rather easy thing to do. This is a nontrivial lesson. One can expect that the cost and speed of a given computation will decrease over time, but one cannot expect that the computation will become simpler.

STANDARDS FOR 3D RENDERING

In the past year or two, candidates for standardized rendering libraries have emerged. For example, the GL only exists on SGI machines; however, an initial version of a vendor-independent GL, known as OpenGL, has been developed.[1,2] Another example of a vendor-independent rendering library is PEXlib.[3,4]

The comparison of one rendering library to another is less significant than the trend toward standardization, which can be expected to continue.

CONCLUSION

"Three-dimensional rendering for the masses" is a reality, and it is now easy to make this technology part of a larger computation. In sum, we now have a new, simpler way of doing computations that would previously have required a long-term, large-scale effort.

REFERENCES

[1] OpenGL Architecture Review Board, *OpenGL Reference Manual*, Addison-Wesley, Reading, MA (1993).

[2] OpenGL Architecture Review Board, *OpenGL Programming Guide*, Addison-Wesley, Reading, MA (1993).

[3] Graff, M., *PEXlib: A Reference Manual*, Prentice-Hall, Englewood Cliffs, NJ (1994).

[4] Womak, P., *PEXlib: A Tutorial*, Prentice-Hall, Englewood Cliffs, NJ (1994).

THE AUTHOR

MATTHEW R. FEINSTEIN received B.S. and M.Eng. degrees from Cornell University in engineering physics in 1970 and 1971, respectively, and a Ph.D. in applied physics from Yale University in 1976. He is a physicist in the STW/ASUW Missile Systems Group in the APL Fleet Systems Department and a member of the Senior Professional Staff. Since joining APL in 1978, Dr. Feinstein has worked on variational techniques in scattering theory, measurement and modeling of polarization aberration due to radomes, and development of multiport monopulse radar algorithms.