Evaluation of Visualization Software

Al Globus, Sam Uselton

Report NAS-95-005, February 1995

Computer Sciences Corporation at NASA Ames Research Center[1]

Abstract

Visualization software is widely used in scientific and engineering research. But computed visualizations can be very misleading, and the errors are easy to miss. We feel that the software producing the visualizations must be thoroughly evaluated and the evaluation process as well as the results must be made available. Testing and evaluation of visualization software is not a trivial problem. Several methods used in testing other software are helpful, but these methods are (apparently) often not used. When they are used, the description and results are generally not available to the end user.

Additional evaluation methods specific to visualization must also be developed. We present several useful approaches to evaluation, ranging from numerical analysis of mathematical portions of algorithms to measurement of human performance while using visualization systems. Along with this brief survey, we present arguments for the importance of evaluations and discussions of appropriate use of some methods.

1. This work is supported through NASA contract NAS2-12961.

Introduction

Visualization software is becoming widely used as shown by the large number of visualization environments available today. To gain benefit from using such a system one must be able to distinguish between aspects of the visualization which reflect the data and aspects due to the visualization process itself.

Is the visualization valid? That is, does it show what the user intended? Is the visualization software verifiable? That is, can one demonstrate that the software does what the programmer intended? And how can one evaluate a system's capabilities and compare it with other systems? For the most part, the information necessary to answer these questions is not generated or, if it exists, is not public. Application scientists have sometimes addressed these questions [e.g. BUNI88], but mainly for their own software and application. Papers on the error characteristics of visualization techniques [e.g. DARM95, MARC94] are beginning to appear. Experiments using human subjects to get (relatively) hard data on visualization system performance are required for clinical use in medical fields [VANN94] and have begun for various other applications [e.g. BRYS95]. But much more work is needed.

This paper is a brief look at some of what has been done, and some of what needs to be done to ensure that scientific visualization software is valid and verifiable. We examine standardized test suites, the error characteristics of the visualization process, and human experimentation to test the basic hypothesis of visualization: that visualization improves insight and task performance.

Standardized test suites

One approach to verification and evaluation is to run visualization software on standard test suites that, by general agreement, cover the important characteristics of certain classes of data. Different test suites are necessary for different problem domains; computational fluid dynamics, geophysical simulations and radiological data, for example, present different challenges to visualization systems and therefore need different test suites. One can (and should!) use real scientific data sets to evaluate accuracy and performance, but these data sets are not sufficient. They were designed to investigate phenomena, not evaluate software. Thus, correct software behavior is difficult to verify and the various aspects of performance are difficult to asses. Verification suites and benchmarks specifically designed to test visualization systems would be of great value.

Correctness (the software does what it claims) and accuracy (it maintains appropriate precision) are vital for acceptance by critical users and become more important when visualization software provides quantitative data (e.g. volume of a tumor or clearance between moving parts). Other than blind faith, there is no reason to believe the results from visualization systems are more than approximately accurate most of the time. For example, one well known (but here nameless) visualization package uses Euler integration for streamlines, a technique well known to produce incorrect results in vortices! Worse, there is no accepted way to quantify error, which is always present to some degree. There are differences in the results using different packages, but no straightforward means of measuring how much the results vary, which is nearer the correct value, or if either is even close.

Furthermore, developers testing systems have no well established methods of determining the correctness of systems. Each development team creates its own ad hoc testing techniques.

One may compare the performance of different packages on data sets from various application fields, but the results only apply to data with nearly identical characteristics. A well designed set of benchmarks will support independent characterization of various aspects of visualization package performance (e.g., very fast streamlines but slow isosurface generation). High performance computing has many years of experience designing and examining the results of benchmarks [e.g. BAIL94]. One could do worse than follow their lead.

Consider computational fluid dynamics (CFD). Correctness is particularly difficult in CFD visualization since computational grids often contain complex topologies including multiple, interpenetrating 3D blocks with some embedded points marked as not valid. Visualization software becomes complex to avoid computations using the masked data points and to perform properly in the presence of singularities, where an edge of a grid cell may have length of zero. Other applications have different obstacles.

CFD data sets are legendary for their size. The current state of the art in CFD simulations produce solutions over multiple three dimensional grids defined by several million points. Each grid point is represented by three floating point numbers (for location), and five floating point numbers per node are needed to represent the solution, resulting in tens or hundreds of megabytes to capture an instant of flow. But researchers are simulating unsteady flows, requiring a few thousand time steps. Current visualization environments have trouble with these large CFD data sets. As a general rule, limits on the size data set to be visualized are determined by the hardware available to the user. Developers may be able to give a minimum recommended configuration, but they cannot know what maximum to expect, and can not test all possible data set sizes. A set of variable sized benchmarks could provide a means for developers to discover and communicate the size limitations of their products.

A good set of test data should have at least the following properties (in no particular order):

1. The data sets should be scalable to allow tailoring for hardware configurations and the time available for performing the tests.

2. Results should be easily, visually recognizable as correct or incorrect. Furthermore, quantitative results should be easily interpretable.

3. The variety of grids should encompass normal cases and all known pathologies.

4. Fields defined over the grids should also encompass normal cases and all known pathologies.

5. Grids and fields of all the relevant dimensionalities should be represented.

6. The suite should be easily distributable to most hardware and software platforms.

7. The size (in bytes) of the distribution should be minimized. In particular, it is unlikely that data sets themselves should be distributed, but rather the code to generate them.

8. It should be possible to run the suite in reasonable amount of time.

9. The results should give unambiguous performance (as well as correctness) information.

10. The suite should challenge current systems.

In addition to test data sets, there is a need for a standard set of tests of particular visualization techniques and functions.

We examine a few ideas along these lines for techniques commonly used in CFD visualization:

Integral Curves (particle traces):

* Trace forward in time from a point, then trace backwards in time for the same total time from the end of the first trace. The distance between the start of the first trace and the end of the second is an error measure. Do this for many points in vector fields with various properties. Be sure that some of the traces pass near critical points of various types (saddles, nodes, etc.). Also, force traces through areas of rapid change.

* Change the grid system in a variety of ways without changing the vector field defined over the gridded space, and compare traces starting from the same point. A constant vector field is useful here. Be sure to force traces across grid boundaries and near grid singularities.

* In multi-zone data sets, start traces near grid boundaries and stop just after passing to the next grid zone to measure the time to pass between grid zones. Be sure to include a case with a grid singularity near the transition.

Integral Surfaces [HULT90]:

* Force surfaces around saddles. This is difficult because the surfaces tend to tear.

* Integrate surfaces into vortices. This is difficult due to twisting of the surface.

Isosurfaces:

* Generate isosurfaces on a field where all isosurfaces are sets of spheres. Change the grid such that isosurfaces must be generated for all marching cube [LORE87] cases.

Cutting Planes:

* Examine results in overlapping grids to see how visualization of data in the overlap is handled.

* Include benchmarks for time dependent data with static grids. This will reward the developer who reuses rather than recomputes values when possible. For example, when the location of the cutting plane remains constant (but the scalar field changes), the vertices and the interpolation factors needed don't change.

Interpolation:

* Design benchmarks to test interpolation time and precision, including non-linear interpolation.

Vector field topology [HELM91, GLOB91]:

* Place critical points in grid cells with grid singularities.

* Place critical points on computational boundaries.

These ideas are offered as initial suggestions, not final solutions. There are certainly other needs for other applications, and there are probably additional or even better tests for CFD data as well.

Error characterization

Visualization, particularly of numerical results, can be thought of as constructing models of models of models. Even when visualizing experimental data the concept is the same, only the first few layers of models are different. Each model is an imperfect representation of the system from which it is derived, so error is introduced.

Consider the models constructed in the course of studying a physical system numerically with visualization in the loop.

	Model	Error
insight	Physical system	
simulation science	Continuous math	simplifications, lack of knowledge
	Discrete math	discretization, approximations
	Software	round off, bugs
visualization	Data transformations	truncation, interpolation, resampling,
	Continuous vis' model	interpolation
	Vis' algorithms	linearization, approximations
	Display effects	screen space interpolation, jaggies
	Mind's eye	perceptual, cognitive effects

The first step in numerical study of a physical system, say airflow over a wing, is selecting a continuous mathematical model, in this case the Navier-Stokes equations. In their usual form, these equations presume an ideal gas, but air is not exactly an ideal gas. The scientist must understand the error introduced to ensure that it does not invalidate the desired results.

Similarly, to solve continuous equations numerically on a computer often requires a discrete mathematical model. A given continuous mathematical model may be discretized in several ways, and at a variety of resolutions, each introducing different amounts and kinds of errors. Software implementing the discrete model introduces bugs, and execution of the software introduces round-off errors. All of these error sources are discussed in numerous textbooks, college courses and research articles. Scientists have substantial resources available to help understand the error and ensure that the simulation is faithful to the important aspects of the original physical system. Indeed, a researcher is expected to understand these issues and normally cannot publish in reputable journals if the sources of error are not directly and successfully addressed.

The story is quite different once the realm of visualization is entered. Even before the data reach a visualization system, several transformations may be applied. For example at one facility, CFD data are routinely truncated from 64 to 32 bits before visualization. This is usually acceptable, but occasionally causes problems for the unwary. Similarly, data are often transformed from cell centered to the point centered schemes used by most visualization systems. These transformations introduce error, which must be understood to ensure it does not corrupt the analysis.

Some visualization techniques assume continuous data fields. These fields are simulated using interpolation between the points where data are available. Trilinear interpolation of hexahedral cells is a particular favorite. These interpolation schemes are seldom consistent with the interpolation used by the simulation software, introducing additional error.

Each visualization algorithm introduces its own set of errors. For example, particle tracers accumulate error since each point is generated from the previous point. Different particle tracing techniques will occasionally generate quite different results [KENW93].

Another set of errors come from rendering. When a shaded polygon is rendered using the usual graphics shading techniques, interpolation is carried out in screen space. As a result, the color of a point on a surface is affected by the current viewing transformation! This may be acceptable for qualitative assessment, but the careful user must be aware of such problems.

Better understanding of the numerical error in the individual processing stages is not good enough. How the various errors compound, cancel or otherwise interact is also important but little investigated. Changing the order in which operations are performed can cause significant differences. Interpolating between grid node colors (as done by fast rendering hardware) often gives a different result than interpolating data values and then mapping to colors. To make a careful choice of appropriate visualization methods, a scientist needs the error characterization information. Since scientists will be investigating new phenomena, and applying visualization tools in unexpected ways, they must have access to the error analysis, not just its conclusions.

It behooves the visualization developer to understand the sources of error in the visualization process, thoroughly document the errors -- in the visualization itself if possible -- and take steps to ensure that the magnitude of the error is much less than the errors introduced by the modeling process. After all, users want to see their data, not visualization system induced error.

Finally, perceptual and cognitive effects color an investigator's view of the pixels displayed on the screen. Hopefully, the user's mind's eye will help generate insight into the original physical system, spawning new hypotheses and experiments. How can we tell whether what is perceived illuminates the scientist's questions?

Experiments with human subjects

The grand hypothesis of the visualization community is that scientific visualization improves human insight. Several methods have been traditionally used to prove this hypothesis:

Proof by repeated assertion

Proof by vigorous gesticulation

Proof by pretty picture

(For extended comments relevant to Proof by Pretty Picture, see [GLOB94].) There are better approaches to proving the insight hypothesis. Most visualization practitioners have accumulated a great deal of anecdotal evidence. Such evidence could be, and should be, collected and carefully examined. Better yet, if one wants to prove that the state of a human being has changed, why not run a controlled experiment and measure the change of state?

Ideally, one measures the insight of an individual, shows them a visualization, measures the insight again and studies the differences. Unfortunately, there is no general way to measure insight. However, good teachers write test questions to reveal a student's insight (or lack thereof), so it may be possible in certain specialized circumstances.

Although insight is difficult to measure, task performance can often be measured with some accuracy. Thus, experiments to determine the efficacy of visualization systems and techniques for particular tasks can, and should, be developed.

Although exploratory scientific visualization is difficult to break into specific tasks, engineering and medical visualization applications are fairly task specific and amenable to an experimental approach, as are other analytical visualization uses.

It should be noted that experiments comparing visualization systems are easier to undertake and interpret than experiments to evaluate or characterize a particular system. We therefore focus discussion on experimental comparison of visualization systems.

Consider comparing two visualization systems for task performance. In theory, each system might be characterized as a point or region in a large multi-dimensional space. Although some dimensions can be given, e.g., techniques available, data formats supported and memory usage, not all the relevant dimensions are known or easily measured. Therefore, we have an unknown, probably non-linear function *f:system -> task-performance*. Each domain/range pair of this function can only be discovered by laborious, controlled experiment. Interpolation is fraught with danger, and extrapolation ridiculous. Obviously, characterizing this function -- whose domain is not thoroughly understood -- is a daunting task. However, at present we have almost no experimental data points at all. Even a few widely space, but reasonably firm, data points in a single context and widely available would be of considerable value.

In particular, one would like to predict performance for other, related tasks and predict the effect of changes to the system(s) under study. As mentioned above, predicting performance of tasks other than those studied is difficult. Predicting the effect of changes in the system is also difficult, in part because different aspects of systems interact to affect end user performance.

A final word of warning for those considering controlled experiments of visualization systems, be careful to choose experimental subjects from the target user audience. For example, if insight into moderately experienced users is desired, don't use programmers or novice users (or freshman psychology students) as experimental subjects.

Conclusion

The quality of a visualization system, method or a particular visualization, is almost an unknown concept. System comparisons are based on ease of use, flexibility, speed of operations and the presence or absence of particular features. None of these criteria are really relevant unless one can rely on the images produced to reflect the data without distortion or confusion. After all, it's easy to make *wrong* pictures at sixty frames per second.

Some of the work needed is just the direct application of numerical analysis techniques and dissemination of the results. Other important research will involve fundamental work on identifying all the relevant parameters of "visualization quality" and their relationships. Then the scattering of points in this space where the "quality" function has been evaluated may allow us to see trends or unexplored combinations of parameters.

Acknowledgments

This work was performed at the Numerical Aerodynamic Simulation Systems Division, NASA Ames Research Center under NASA contract NAS-2-12961. The authors have benefited from discussions with many visualization researchers, developers and users, but especially with Geoff Dorn of ARCO Production and Technology, Joseph Gitlin of Johns Hopkins University Medical Institutions, Michael Vannier of Washington University School of Medicine, and the audience for our Visualization '94 panel, and with Steve Bryson and many other colleagues at NASA Ames.

References

[BAIL94] D. Bailey, et al, “The NAS Parallel Benchmarks (94),” RNR Technical Report RNR-94-007, NAS Division, NASA Ames Research Center, March, 1994.

[BRYS95] S. Bryson, S. Johan, A. Globus, T. Meyer, C. McEwen, “Initial User Reaction to the Virtual Windtunnel,” AIAA95-0114, 33rd AIAA Aerospace Sciences Meeting, Reno, NV, January, 1995.

[DARM95] D. L. Darmofal, R. Haimes, “An Analysis of 3D Particle Path Integration Algorithms,” (to appear) AIAA CFD Conference, San Diego, CA, June 1995.

[GLOB91] A. Globus, C. Levit, T. Lasinski, “A Tool for Visualizing the Topology of Three-Dimensional Vector Fields,” *Proc.Visualization ‘91*, IEEE Computer Society, San Diego, CA (1991).

[GLOB94] A. Globus, E. Raible, “Fourteen Ways to Say Nothing with Scientific Visualization,” IEEE Computer, vol. 27, no. 7 (July 1994), pp 86-88.

[HELM91] J. L. Helman, L. Hesselink, “Visualizing Vector Field Topology in Fluid Flows,” IEEE Computer Graphics & Applications vol. 11, no. 3 (May 1991), pp 36-46.

[HULT90] J. P. M. Hultquist, “Interactive Numerical Flow Visualization Using Stream Surfaces,” *Computing Systems in Engineering* **1** (2-4) pp. 349-353.

[HULT91] J. P. M. Hultquist, personal communication.

[KENW93] D.N. Kenwright, Dual stream function methods for generating three-dimensional streamlines, Ph.D. thesis, University of Auckland, New Zealand, August 1993.

[LORE87] W. E. Lorenson, H. E. Cline, “Marching Cubes: a High Resolution 3D Surface Construction Algorithm,” *Computer Graphics*, Vol 21, No 4 (July 1987), pp. 163-169.

[MARC94] S. R. Marschner, R. J. Lobb, “An Evaluation of Reconstruction Filters for Volume Rendering,” *Proc.Visualization ‘94*, IEEE Computer Society, Washington, DC (1994).

[VANN94] M. Vannier, Position Statement for Panel: “Validation, Verification and Evaluation,” published in *Proc.Visualization ‘94*, IEEE Computer Society, Washington, DC (1994).