There is no doubt that visualization is very useful, by enabling people to understand that masses of data and information otherwise hidden inside of computers. However, after many years developing visualization systems I have to confess to skepticism about many of the hottest (i.e., coolest) visualization ideas.

Virtual Reality

When I started at the Space Science and Engineering Center (SSEC) in 1978 they were already experimenting with red-green stereo viewing of pairs of images from satellites over the eastern and western United States. This gave viewers a qualitative feel for the altitudes of clouds. While these displays were very interesting, they never became the basis for serious work because scientists were not prepared to make quantitative judgements based on their depth perception. Instead, they developed algorithms for estimating the altitudes of cloud tops from these image pairs. Their primary serious use of visualization was to pick likely candidates for their automated cloud-tracking algorithms, and to check the quality of the resulting wind estimates. Those tasks required animated 2 D images, but not 3 D.

Nevertheless, the search was on at SSEC for useful 3 D visualizations. We attended the annual Siggraph conferences, studied Foley and van Dam’s book, experimented with a variety of rendering techniques that ran overnight on IBM mainframes, and experimented with cross-polarized stereo displays of our 3 D rendered images. Those experiments indicated that non-interactive
3 D was not very useful to scientists, except to produce videos that they could show to other scientists. The first useful result was the Vis5D software in 1988, which exploited the interactive 3 D graphics of the Stellar computer to help scientists understand the behavior of their complex algorithms for simulating the atmosphere.

Head-mounted virtual reality displays had serious problems with latency in response to head movements, so we never thought seriously of applying them to Vis5D. But then the Electronic Visualization Laboratory's CAVE offered an alternative that reduced the latency problem, so we adapted Vis5D to Cave5D for the VROOM at the Siggraph '94 Conference. The results were very nice. Figure 1 shows an image from our Cave5D demo at the Supercomputing 95 Conference. Cave5D is still one of the most widely used software systems for the CAVE and ImmersaDesk, thanks in part to improvements made by Glen Wheless and Cathy Lascara of Old Dominion University. However, this experience indicated to us that immersion does not add much value beyond interactive 3 D workstations, but is many times more expensive. Furthermore, if scientists have to go to a special room to use a piece of equipment that is not that much better than the equipment in their office, they'll stay in their office.

Based on my experiences since 1978, I am skeptical that immersive virtual reality will be of any real benefit to the scientists I work with any time in the next ten years. I think that large-scale use of immersive VR by scientists will not come until costs are low enough to put it in every office, the latency problem is solved, and the need for special glasses or helmets can be eliminated or at least reduced. These problems will eventually be solved. Of course, VR is useful now for flight simulators where it is important to simulate a real experience and where cost is not important. Furthermore, in a flight simulator the scene is far away from the viewer, so that stereo glasses and head motion detection are not necessary.

**Volume Rendering**

The 1988 Siggraph paper by Hanrahan, Carpenter and Drebbin on volume rendering was a great turning point in graphics, showing the world a strikingly beautiful new way to depict numerical data. At SSEC we reacted by experimenting with volume rendering on the IBM mainframe and presenting a paper about it at the 1989 Volume Visualization Workshop. The point of that paper was that volume rendering could be approximated fairly well using ordinary polygon rendering, but required N cubed polygons for a N x N x N volume as opposed to order of N squared for iso-surface rendering (assuming
reasonably smooth data). My belief that 3 D graphics had to be interactive to be useful to scientists led to my first skepticism about volume rendering.

However, by the mid-1990’s 3 D graphics hardware was getting fast enough for interactive volume rendering of moderate sized volumes. Scientists at the U.S. EPA asked us to add volume rendering to Vis5D, which we did using the polygon approximation. Figure 2 shows a nice volume rendering of vorticity as computed by a weather model. Volume rendering is only one of the different ways that Vis5D can render scalar fields like vorticity. Others are shown in Figure 3, including iso-surfaces, contour lines on plane slices, and colored plane slices.

After we added volume rendering to Vis5D, we observed that it was rarely used by scientists. The primary reason they give is that volume rendering is not as quantitative as other techniques. I recall hearing Henry Fuchs say something similar to this during a presentation to one of the Gigabit Networking meetings, based on his experience with medical users.

Considering that volume rendering is the only technique that comes close to realistic rendering of clouds, it is surprising and interesting that atmospheric scientists prefer iso-surfaces and plane slices.

Thus, while I think volume rendering should be among any visualization system's set of tools as a means of producing beautiful images, I am skeptical of its importance to scientists in their daily work.

Visual Programming

Data flow visual programming was a very hot idea in 1989, and a number of commercial data flow visualization systems appeared on the market at that time. It was an approach to the very important problem of finding a way for non-experts to customize their visualizations. The idea was that the user has a palette with visual icons representing various basic visualization operations, such a “read a file”, “compute an iso-surface” and “render a geometry”. Users graphically arranged these icons into networks that implement their application. There was pressure on me to rewrite Vis5D based on one of the data flow systems, including gentle pressure from SSEC's Director and the clear preference of funding agencies for work based on data flow systems. However, I was skeptical. These systems made it difficult to optimize memory use and computing time. They made it difficult for user
interactions with renderings to flow back up the data pipe. And their large modules could not be mixed in with the grungy code involving integers and booleans that are necessary in any complex program. This was the basic problem: data flow was very appealing for simple examples, but was more of a hinderence than a help for complex real-world programs.

By now the organizations that created the data flow visualization systems have augmented them with ordinary scripting languages for invoking their modules, and in some cases replaced them completely with object-oriented libraries. In other words, they are now skeptical too. So far, the only way to help non-experts customize their visualizations is to give them good visualization libraries.

Of course, there will always be someone offering a tool that enables non-programmers to write programs. But it won't really be possible until computers are intelligent enough to write programs, which won't happen anytime soon. Rather than disappearing, programs are becoming larger and more complex. This has been enabled by better programming tools. The real needs of programmers are not addressed by visual tools for connecting a few modules together in a simple program, but by visualization tools to understand the structure and behavior of complex programs.

Things that I'm not Skeptical About

There are a number of ideas that are not directly visible in displays but nevertheless have been important to the success of visualization. Some of these relate to how data and information are exchanged. For example, the development of scientific data models and file formats has made it much easier for users to get their data into visualization systems. A critical moment was the workshop at Siggraph 1990 on data structure and access software for scientific visualization, organized by Lloyd Treinish. Attendees included people who played leading roles in IBM Data Explorer, HDF, netCDF, DOE ASCI CDM and VisAD.

The Unix operating system (as well as X Windows and OpenGL) has been important for making it easy to port visualization software between graphics workstations produced by competing companies. That helped make visualization more affordable.

I'm not skeptical about Java, and that probably puts me in the minority. Java has certainly had a lot of hype, and there have been some problems on the road to true platform independence. However, as software becomes more complex it becomes more imperative to have a technology that enables computers to exchange programs over the
network. ASCII, HTML, Postscript, PDF and Word enable computers to exchange text. GIF, JPEG, TIFF, RGB, PNG and many other formats enable the exchange of images. MPEG and others enable the exchange of animations. VRML enables the exchange of 3 D geometry. XML is designed to be extended to various kinds of information. But the most important type of information in computers is programs. Java is the format that enables computers to exchange programs. This strikes me as a fundamentally important capability.

Distributed object technology is a way to organize software across networks. It makes it possible to design network protocols at the application level simply by defining method signatures. This will become important as networking becomes part of all software designs. I am not skeptical.

Finally, I am not skeptical about collaborative visualization. Just as the utility of 3 D graphics had to wait for fast 3 D graphics hardware and mature 3 D visualization software, the utility of collaborative visualization must wait for faster networks and mature software.