SOFTWARE DIRECTIONS FOR SCIENTIFIC VISUALIZATION

Chair: Gordon Bancroft, NASA/Ames Research Center

Panelists:

Roy Hall, Wavefront Technologies
Mike Kaplan, Ardent Computer
Al Lopez, Apollo Computer
Alvy Ray Smith, Pixar

SOFTWARE DIRECTIONS FOR SCIENTIFIC VISUALIZATION

Introduction, Panel Chair: **GORDON BANCROFT**

(slide #1)

Let's get started. The title of the panel is "Software Directions for Scientific Visualization." First, to get us all thinking along the same lines, here's my "Webster's Dictionary" definition of scientific visualization.

(slide #2)

What I've done here is define what I feel to be some immediate and current issues surrounding software directions in scientific visualization.

Let's begin with standards. Obviously, visualization standards are important, especially with regards to data formats. Software standards are also an ongoing issue, and perhaps we'll hear some discussion about this later. This industry is changing so quickly that efforts to standardize have not been able to keep pace. There have also been problems with standards in scientific visualization areas with regards to extensiblity.

The next issue is one of "modeling the environment versus modeling the model." The follow-up slides show some examples of what we do at NASA/Ames Research Center. And I think you'll agree that what we're after is the environment *surrounding* the model. This has an impact on the next point: scientists are not after photorealism as an end goal. A scientist is interested in representing data. Maybe if there were no resource penalties for using, say, ray tracing we would use it. But, after all, in a photorealistic world you can't see airflow, or pressure, or density... and that is precisely what we are trying to model at Ames. The problems of visualizing data are entirely different than the problems involved in visualizing "photorealistically."

Here is a more practical example of what I am talking about: I showed a Gauraud-shaded computational model of a space shuttle to one of the scientists. He'd been used to looking at wire frame or flat shaded models and said, "What happened to my grid?" To a mathematician or a scientist, it's an issue of representing scientifically derived information. Shading that is *basic* to photorealism may actually detract from visualizing scientific data.

Other issues we really have to start looking at are 3-D techniques and stereo. We don't have a good way to interact in 3-D. We've been doing some work with stereo, which goes along with 3-D techniques, and found that to be a very effective way of modeling fluid dynamics.

(slide #3)

This next slide is what I consider current/future issues. I think these are things that we need to look at after we solve some of the problems that I presented on the last slide.

Volumetric visualization is a relatively new buzz word in the field of computer graphics. I think the other panel members will be talking about different ways to do this. Basically, the question is, Do you use fast geometric processors to draw arbitrary planes in a volume of data, or do you do that with a frame buffer and some other method where you move the scene around

and move orthographic planes through that? And that's coupled with the next question, Are we going to draw things in immediate mode? That is, we're not going to have display list memory anymore? Are we going to have to have software that just draws stuff on the fly because that's the fastest way to do it? And that's related to the earlier panel and their discussions about the directions of hardware.

Distributed graphics processing and distributed processing is a very important issue. Coming up with ways to be able to drive scientific analysis from workstations using distributed graphics is something that we have to start looking at seriously; we're doing that at NASA/Ames Research Center.

In the earlier panel, before this one, the subject of video output and input came up. What I'd really like to bring up here is the issue that, since we have a digital workstation and we have digital video and these formats are defined, a software solution exists to complete this path. We need to be able to get digital video from the workstations. (Perhaps the widely accepted D2 digital video format from Sony and Ampex) And it doesn't have to be stored. Tom Jermaluk, from Silicon Graphics, was talking about having to store NTSC in memory, but it might be enough to just produce it. Then it could be stored on outboard equipment, like an Abekas A62 digital video disk/frame store, for later editing and viewing.

And then finally, just to make a mess out of everything, once we figure out how to do steady data, we've got to figure out how to do unsteady data. Unsteady simulations experiment in some 4th dimensional domain, perhaps time. Consider an aircraft that is being simulated at a given airspeed, angle of attack, etc. If you take that airplane and move it (increase the angle of attack) or change the geometry (move a flap), then we are calculating an unsteady phenomenon. The problems surrounding data management and scientific visualization of these results is tremendous.

I'm just going to run through some slides here that are examples of scientific visualization at the NAS (Numerical Aerodynamic Simulation) facility.at NASA/Ames Research Center.

(slide #4)

Here is the shuttle with one side rendered using computational results, and the other side rendered with the wind tunnel results. The quantity being displayed here is total pressure on the surface of the shuttle with it's SRB's (solid rocket boosters) and external tank attached.

I'd like to be able to shade this model, and I don't want to do that in RGB space. I want to do that in color index space, or have a user definable range of colors. I don't know of a single workstation available today that will do this in real time.

(slide #5)

Here is an example of a rotor stator analysis. We call this hot streak where we inject hot gases into a turbine and watch the way they mix downstream.

(slide #6)

Here's a jet in ground effect with particle traces. Particle tracing is a method that we use a lot.

(slide #7)

Here's an F-16 with particle traces again.

(slide #8)

And here's a rotor stator again, 3-D, with pressure mapped onto the blades.

Okay, that's the end of my introduction. I'd like to introduce our first speaker who is Al Lopez from Apollo Computer. Al.

AL LOPEZ: Good morning. I'm going to be talking about network computing and scientific visualization.

Network computing is a relatively new term under the umbrella of distributed processing. Generally, when we think distributed processing, we think about data communication protocols, file access and file transfer protocols. Network computing, on the other hand, tries to address the problem of getting the processing out to the data or to the computing resource itself.

I'm not an expert on scientific visualization. However, I've attended some conferences and found the recurring theme in many of the papers which I've read that indicate that one of the problems people are dealing with in this area is that they're flooded or actually inundated with torrents of data. There's data just about everywhere on the network at many different locations.

The problems dealing with this data are: How do you process it? Where do you process it? Do you move the data around the network? Or do you try to position it strategically next to the computing resource that's going to be processing the data? Then, How do you get the computing or that part of the application that is going to operate on that data out to that resource?

Here's a very simplified model of the scientific visualization environment. You generate raw data through either simulation or acquisition, analyze the data, generate more data, and process more data. And,using visualization techniques, you look at either the raw data or the processed data through imaging technology or graphics. That can also generate more data, like geometry data which then gets rendered to generate image data.

One of the effects that scientific visualization has had is to provide a higher bandwidth of information to the scientists. This bandwidth in turn gives the scientist greater insight on analyzing the data. And that insight produces intuition. What a person wants to do with that intuition is sort of follow their gut feel. And as a result, they want to control or steer the components of the application, like the acquisition, the analysis, and the visualization, and thereby, you have a desire for more interaction.

Now I moved this slide ahead earlier, but basically, by adding steering into that model, we've completed the model and now have the ability to control the visualization analysis and simulation. However, there is a very key point that needs to be made here. This is no longer a batch process. You can't run a simulation and then some months later run some analysis on it that takes a significant amount of time and then spend a couple of weeks generating visuals to further analyze that data through visualization.

So basically, what we've done by introducing steering into this model is brought these pieces in what needs to be a very interactive application. The ways that this application needs to become interactive also generates a demand for higher bandwidth communications and also for greater compute speed.

Now there are some solutions to this. You can take all of these components, put them into one application and run it on an individual's personal Cray class computer. Well obviously, there's not a big market, or at least we haven't found one for multi-million dollar single user

workstations. And in fact, the opposite is quite true. What we're finding is a requirement for low end systems that have an easy-to-use interface.

Let's look at the environment where scientific visualization applications run . It's a networking environment made up of local and wide area networks and all types of heterogeneous computers, from supercomputers down through mainframes, mini, super workstations and PCs. How can we make an application that's running in this environment interactive?

Well, one approach is the brute force approach. You just add more computing power and add more communications bandwidth. However, one of the problems with this approach is that it is in some ways self-defeating in that the same power that you're adding to analyzing and looking at the data is also being added to generating more data. And what you've done is basically scaled the problem up and created an even larger problem.

Another part of the solution is to take an architectural approach to making that environment interactive through placement allocation and distribution. You want to place the data in strategic locations and then move the computing resources out to where the data is and where the computing resource is. You want to allocate resources so that the environment is load balanced enough to provide almost dedicated resources to the user. In effect, what you want to do is give an individual user multiple computers at their disposal. And you want to do this through a method of distribution that really harnesses the compute power that's in the network.

Let's look at a taxonomy of the scientific visualization application. The role of these components, which we've already discussed, all of which generate the data that we've discussed, what you'd like to do is take that application and move it into that environment and distribute the pieces around the network. You'd like to put the data simulation, perhaps, on a supercomputer class machine, acquisition on a mainframe, and some of the visualization or any of the algorithms that are highly parallelizable out on a network running on multiple processors simultaneously.

However, there are two key criteria that need to be considered in doing this. The first one is, you want to do this in a way that is transparent to the application and to the application developer. You don't want to add the burden of dealing with the network and all of the communications protocols and data access protocols to the job that the application developer already has to do. You want to provide a model that works the same locally as remotely.

The other thing that you need to provide is application immunity to changes in the network. If you change the placements of data on the network or the allocation of resources on the network, you don't want to go back and change the distributed computing application in order to adjust to those changes. You really want it to be immune to those sort of changes.

Now network computing solves both of these problems, transparency and immunity, in the following ways. First of all, it's based on a remote procedure called paradigm. Here the calls that are being made to remote procedures have stubs that are provided by the environment which marshal the parameters into packets that get transferred over the network where they get decomposed or unmarshalled on the remote process and turned into a call frame to the subroutine. The subroutine doesn't know if it was called locally or remotely. So therefore, you have the transparency both on the client's end and also on the server end. That in and of itself, however, is not enough because you don't want people to have to write these stubs and have to deal with the remote procedure call mechanism

So another component of the environment has to be a description language, a network definition language, which we call NIDL in NCS, which basically does that for you. It's a declarative language external to the application. You run it through a compiler. It generates the

stubs. The stubs get bound into your application. And all of the handling of those packets and marshaling and unmarshalling of procedure arguments are handled by the remote procedure call environment or the network computing environment. This doesn't imply that these things now have to run remotely. In fact, this works just as well locally as it does remotely. And that's another important criteria.

The other aspect that's very useful and provides a lot of flexibility is taking an object oriented approach. You'd like to be able to think in terms of operations performed on different types of objects. A good example of this is, if you have a program like "ls" in UNIX which lists a directory listing of files, you'd like to be able to use that same code and reuse it without modification if what you're asking for is the contents of a directory, or if you're asking for the contents of a print server queue.

In that case, each of those are of a different type and all you need is to be bound or given a handle to different methods which know how to extract the contents from either directories or print server queues. And this is done in NCS through a client server broker model. The broker basically has the information about where the servers are, what operations they support, and where they live and how to give you the handle, give the client the handle to access the right set of methods to perform that operation on that type of object.

Now we've talked about things which give you transparency and things which make it automatic for that transparency to exist,like the NIDL compiler, and the flexibility of object orientation. But we haven't talked yet about immunity.

The immunity needs to be provided by having a data base of information in the environment which keeps track of where the servers live once they've registered themselves, and if they get moved, where they've been moved to. So you have the new location. And then the client needs to access that information through an interface which we call the location broker interface.

Now this again gets handled by the codes that get generated for the stubs and the application developer does not have to worry about that. As we move resources around the network, the location broker data base is updated and the client, the next time it runs, simply asks for that capability, that set of operations, on a particular type of object. The broker resolves that request and returns back a handle with which you do the bind to the appropriate set of methods in the distributed environment.

Now we've talked about distributing applications. I want to talk just for a couple of minutes about distributing graphics and distributed graphics systems. I think that distributed graphics is really in its very early stages right now. We have models of distributed graphics built around client server protocols like X-windows, NeWS and PEX.

We also have some systems that do distributed graphics, in fact some of which use NCS, like the intelligent light package and our own ray tracer Apollo, as well as many packages that are doing distributed graphics in their own environment, like the Pixar RenderMan which runs on multiple processors. However, we want to generalize this and make it more available in a general purpose environment. And I believe the direction that that's going to go is towards and object oriented graphics system.

An object oriented approach would provide data abstraction. You'd like to, say, rotate yourself 30 degrees and not care about whether it's a bit map image that's being rotated or some geometry that's being rotated, in which case you'd use different algorithms. The application shouldn't have to worry about that. Just issue that command and have the object oriented system worry about making the proper binding.

You'd also like to be able to provide for user extensibility, allow them to add things to the environment which are not currently in the environment and not have to wait for a release from the vendor to support that new feature. So an object oriented environment provides for that.

You'd like to be able to provide distributed access to shared objects that can exist anywhere on the network and also be able to invoke objects that themselves are distributed computing objects. If you say to an object, "Ray trace yourself," have the method for that object to use the distributed computing environment to spawn out copies of the ray tracer on multiple nodes and have that happening in parallel.

And in order to provide all of this, you really need to consider an object programing model, and in fact, there is an object programing model that has been developed at Ardent Computer in the Dore system, which Mike Kaplan will be speaking about next.

I think that in terms of future trends and directions, an object oriented graphics system coupled with an object oriented network computing system, such as NCS, can really provide the framework for distributed object oriented graphics.

In summary -- and I apologize that I don't have a summary slide -- I just want to stress the following few points. First of all, it's not just enough to just move the data faster and process the data faster, not when you've introduced steering into the equation. You really need to support interactive distributed applications. Those application need to be done in a fashion that makes the network transparent to them, and they need to be immune to changes in the network. And the next generation of graphics environments I think will be both object oriented and distributed and of great value in the scientific visualization environment. Thank you very much.

Now, Mike Kaplan from Ardent Computer.

MIKE KAPLAN: Hi. I just want to note that you can tell that we're the software panel because none of us are wearing suits. Everyone in the hardware panel was wearing a suit.

I only have five minutes because I'm going to show you a five-minute videotape that shows some of the work that has been done, using the tools that I'm going to talk about. So this is going to be very fast and I'm going to ask you to visualize the slides that I might have brought.

About 2-1/2 years ago when I joined Ardent, we had a really big problem in terms of tools. We were building a machine which combined mini-supercomputer performance and highly interactive graphics and it was a very closely coupled system and it was not a thin wire system where the graphics data and the computational data were separated.

And we looked around at what was available and decided that none of the models that were currently in use, in terms of standards or commercial software, really satisfied that kind of abstraction that we wanted to present to the user. So we built a system called Dore, which is an attempt to be a stab at this new model of object oriented and distributed graphics. And we believe it's just a first step, but at least a framework in which more progress can be made.

Primarily, the purpose of Dore is to provide an application interface to three dimensional graphics. It is not a rendering system. It is not a modeling system. It is not primarily a graphics data base, and it's not a floor wax either. It's really an application interface to 3-D graphics.

And what that means is that the person who talks to it should not have to understanding rendering, where the processing is going on, what decomposition of geometry is being performed at a lower level, and what attribute assignment mechanisms are being used. In other words, it should be a framework on which manufacturers and laboratories and end users can all

build and provide their users, or themselves, with a consistent user interface to a lot of functionality.

To do this, we felt we had to move the level of abstraction much higher than it had been. So the key elements of the system are, number one, it's object oriented in a very macro level. In other words, it's still efficient. For people who know a lot about Smalltalk and so on, in the past, object orientated at the level of Smalltalk tends to be inefficient for computation. So we decided to do it at a macro level so that primitives and attributes and views and devices and so on are objects. But it's programed in standard C and can be portable to a number of systems.

Number two, it's extensible. The user can provide his own attributes and primitives and other rendering methods besides the ones that ship would be initial system can be plugged in by other vendors, by other computer companies or by the end user.

We don't believe that we know how to solve all the visualization problems and know what all the primitives are going to be needed in the next five years or what kind of visualization rendering methods are going to be used. But we do feel that there ought to be a way that these things can be put together so that you don't have to wait three years for a standards committee to decide on it or two years for a group of manufacturers to decide on it or one year for your favorite vendor, under a lot of pressure from you, to provide it to you. So it's extensible.

It does not contain the idea of a renderer. Dore does not say how you render something. It says you hook in different renderers and they do the best job that they can for the given devices that they are making output for. So for example, we ship a dynamic renderer which drives our real time graphics hardware. It could drive other people's as well. And we ship a high quality renderer that makes nice stills. RenderMan interface could easily be hooked in to generate -- to do rendering from the same data base for people who want to achieve the effects that RenderMan is specifically set up to handle.

Another aspect of this is that we do not enforce a display list or a drawing style in the data base. The data base contains a scene description, relatively abstract, and based on the kind of primitives that the user uses in his own modeling and/or analysis system, and furthermore, it can be closely coupled over a tight interface.

So at this time, let me start the tape. But before we start the tape, everyone can pull out their entry level Ardent system and view it directly if they would care to.

The first example is of a simple constraint base model. Here's the Ardent national flag waving in the wind. All of these animations were made in the last month and a half by people who had never used our system or Dore before. Here's the Siggraph national flag.

But the analysis application or the constraint base model here and the graphics are very closely coupled. The objects are interchanged and the analysis or the data for the meshes and so on are in the forms that the computational program wanted it in.

Here are some examples of a simulated wind tunnel where we are animating particle traces which are computed over the surface of a space shuttle using a panel method, and while we're doing this, we can also be rotating the object so that you can get a feel for what's happening on all sides of it.

The next piece, I believe, shows a 2-D -- okay, this is it moving through the particle fields. This is the next piece that shows a helicopter rotor wash which is analysis of the vorticity of the air coming off of the blue helicopter blades, and we're also viewing that from a variety of angles as multiple frames of data are played back.

Next is a 2-D solution where you see it converging on a solution of flow over a cylinder, and in the next case, over an airplane wing. And as it converges, you'll notice the Bernoulli principle being illustrated. Less pressure on the top.

In this case, we take the photographic data of Yosemite Valley, map it onto a three dimensional height field, actually make a mesh of triangles, color triangles out of that, and then rotate them in real time. Now this kind of object is an object which can be added to Dore if it isn't already there. So if you have a particular kind of primitive that you use in your analysis all the time, then you can add it to the system and get all the benefits of rendering.

This is a solution of a wave equation in an L-shaped membrane, and I believe that the color has something to do with the data associated with it. I really don't understand what that one's about. It's done by a mathematician.

The next one is a projection of four dimensional space into three dimensional space where the color, position and movement are all indicative of elements of that four dimensional space as we're moving through it.

I think the most important thing about these animations is not that they're pretty or that they show a particular area of science, but that they were all done in a month and a half by people who didn't know how to use the system. And on one machine. So there was a fair amount of resource contention as well.

Now this is a presentation, the first time ever, of a rational bi-cubic data spline surface patch work done by Brian Barsky at the University of California. These are like nerves, except that unlike nerves, you can have local control of bias and tension. And we're using this software to visualize what happens when you change each of the four different controllers, the weighting of the nodes, the node spacing, the bias and the tension parameters.

The bias and the tension are additional parameters provided by the data formulation. So you can see, especially at the end, I believe, the surface being pulled tighter and tighter over the control points, but without moving the control points.

These are the first seven vibrational modes of the tuning fork with finite element analysis of the stresses of the fork. I don't think you'd really want to stress a fork this much. It's a rubber tuning fork.

This is what's called a locking cubes method of doing volume visualization in which we're showing the explosion of gas in a rotary engine compartment. So the yellow stuff there is the gas and it's blown up. It's now spreading itself through the inside of the compartment. In the next case, we see the actual mechanism of the Wankel rotary engine moving in response to the explosion of the gas. And finally -- not finally, but almost finally -- in our oil reservoir showing the oil being pumped to the surface by ejecting gas from the surface.

Now this is a temperature simulation of the Atlantic Ocean. The first one is over a six-month period and the next one is over a two-year period. Someone has remarked that if you were extremely patient, you could surf on the temperature gradient on the Atlantic Ocean. Here's the two-year simulation. This is called geologic time surfing.

And finally, this is something one of our students did. He came in to take a class on Dore. He did this on the half a day that he was there for the class. And we put it in just because we were also impressed that somebody in half a day of training could take their data and do this. And here are the credits.

Okay. So my time is up, so I'd like to introduce Alvy. I asked him what he'd like me to say about him and he just said, "Here's Alvy."

ALVY RAY SMITH: Well, I first of all want to say that I very much appreciate Al Lopez's overview. I thought it was quite detailed and instructive, to me at least. Several times he mentioned the difference between geometric data and imaging data, and that's the main point, the main distinction that I'd like to draw today.

This is probably an audience that understands the distinction completely, but I have been surprised over the last couple of months to find scientists unaware of the distinction. So please allow me to hammer in the distinction today as my contribution to this panel on visualization.

The idea is very simple. The key word is geometry. There are two ways to make pictures, or at least two ways, two general ways to make pictures with computers. And one of them is based on geometry. We're all familiar with it. We've done it for years. We take an object or a scene or something and describe it in terms of abstract geometrical concepts, such as line, polygon, patch, curve, nurb, cylinder, sphere, hyperpatch, etc. A geometric concept that at some point gets scan converted into a set of pixels so that it can be looked at.

The other one is where you take samples of some continuum and place those samples directly onto a display, bypassing any notion of geometry, other than the fact the display is square or things like that, or rectangular.

It seems obvious, but I've had people walk up and say, "You mean you don't have to use geometry to put a picture up on a computer screen?" It's like some basic intuition from lay people, at least, non-Siggraphians that computers mean mathematics, and mathematics means geometry if you're talking about pictures. So the geometry must be involved somewhere. You've got to be kidding. You have to go through geometry somewhere.

So, my main message is no, there are a lot of pictures. Perhaps in scientific visualization, 90% of the data does not come from geometric sources, geometrically described objects in the first place. And I'd like to show several examples of those, visualizations of non-geometrically derived data today.

Now the first thing I'm going to do is show some slides to kind of calibrate you on the videotape that I'll show shortly thereafter. This is a puff a smoke, a real puff of smoke that was acquired with a laser beam. The light rays bounce off the laser beam and were captured by a CCD array. So this is an example of a volume filling set of sample points, data points, representing particle densities in three space, and it's also so complex that it would be very difficult to abstract this object with geometry.

This is turbulent water flow. You'll see all these things move in a moment. This is the main one I wanted to put up on the screen because the videotape is not very good. This is the universe. It's a cubed universe about -- I don't know, it says in the videotape a million light years across, or something like that.

A stress analysis throughout the volume occupied by a CAD/CAM designed mechanical part. A mathematical function in three space. I think this is a Mandelbrot set in quaternian space or something like that. And then some medical images that are all derived from CAT scan data. Again, I emphasize that there is no geometry in any of the pictures you've just seen, except for one. I'll let you figure out which one it is. That's terrain rendering, we'll go back.

Okay, good. Would you show the videotape now please? And be ready to kill the sound when we get to the Pixar marketing hype. Please play the video.

There's one other that I tacked on after the Pixar logo that I'd you to see. While we're waiting for that, let me point out the one -- I mentioned that one of those data sets was derived from geometrical data, and that was the stress analysis CAD/CAM part, was derived from a finite element mesh. The stresses were computed at the nose of a finite element mesh and then scan converted into a volume filling set of samples which we then display using our volume imaging techniques.

This is Freddy. This is the first complete CAT scan reconstruction of a human being, showing several different visualization techniques. And I emphasize again that there never was a geometric step at all involved in all but one of these visualizations.

Okay. Would you stop the tape, please? That's all I had to say. The next speaker is Roy Hall, thank you, from Wavefront Technologies. Oh, he's no longer from Wavefront Technologies. He's from Cornell University, as of last week, I believe.

ROY HALL: Good morning. I'd like to talk a little bit about visualization techniques and how they relate to some of the commercial software packages that might be available. I think we're seeing kind of an abundance of entertainment animation packages that are available. And entertainment animation has done a lot to drive commercial availability of packages that let you and I generate images at home on our own little PCs or whatever.

In the realm of scientific visualization, certainly the problem we're dealing with is substantially different than in entertainment. But we are basically trying to communicate information. And I think that some of the tools that are available are useful for communicating scientific information also. But there are certainly a number of hard questions that need to be asked and need to be solved for those tools to become much better.Basically, how do you use the software that's available now to solve the visualization problems that you have? Because most of the software in commercial animation is not oriented towards scientific visualization. Another question is, what's going to be the future of this software? Which way are manufacturers going to orient themselves? And are they going to do products that are going to better suit the visualization needs?

And probably the most important question is, how do I turn graphics and visualization tools into tools that help me look at the real problem I have to solve, rather than making graphics itself a more imposing problem than the thing I'm simulating? And this is probably one of our greatest challenges right now.

So basically, we want to talk about how we use software that has been designed to make pictures like this. And I should have a second carousel there. Well, we'll do it without the second carousel. Basically, how do I use software that was generated to make images like this? How do I use that software to communicate things like this, which is a scientific visualization of some molecular work?

Okay, let's look at what we get normally in an animation package. We have a system that has some scripting tools that allow us to plan motion over whatever time period. We have some modeling tools that let us build geometries and things like that. We have a rendering tool that lets us make an image. We have some compositing tools, or really, some post-processing tools that let us do some amount of image processing, or combining that image that we've generated with text or other types of things. And finally, we have some sort of image output and some tools that let us get it down onto videotape.

Basically, when we go into the scientific realm and we're talking about visualizing data that's created by simulations or by direct measurement, the data that's created becomes a scripting tool and in some cases becomes the model. And at some point we need to take this data, this scripting information, or this model, and pump it into a renderer. So we're really concerned about what the interfaces look like to do that.

One of the things that is very much the word of the day is volumetric rendering. And certainly, we've heard a lot of that at this conference. But I think that we find that things like this are perhaps inadequate to address all of the needs that we have in visualization. This is a simple volumetric rendering of some storm data that was done at the University of Illinois supercomputer center, and I believe the data was generated by Richard Matson.

Whoops, wrong data. This was done by Bob Wilhemson. And basically, we're looking at a storm vortex. We're looking at the density of rain within that vortex. It's a 3-D array of points. It's uni-variate data that changes over time, and basically an implicit surface is generated and normal rendering techniques are used.

These are cosmic strings colliding in the universe. And I have no idea what that's all about, but it's by Richard Matson. And again, it's a technique where we're really using uni-variate data and we're watching it vary over time. And we're getting some insight into what's going on. But I think the problems have become a bit more interesting in scientific visualization, are the cases where our processes are very complex and we have more to look at than a single value that varies over the domain space of our solution.

In this case, we're looking at a test boring and we're checking some of the seismic things that have happened here. I guess they're measuring density and some kind of fracture structure. And we basically, the depth is reflected by position along this test boring. We have an orientation as we go around the test boring. Color tells us something about density, and the depth of the surface tells us something else about the structure. So we're adding more elements to the visualization ... once you get into the multi-variate cases. And I think what you find is that some of the commercial animation systems provide a test bed or a prototyping tool to look at new solutions.

This is an example of a neuron chain that's firing and this was done by Mike Miscagni, and basically, we're watching the propagation of things as they go around the ring. It's a ring of neurons. This doesn't look anything like neurons. I don't have any idea what neurons look like. But basically, what we're trying to understand are patterns that have to do with the interactions of these neurons, and the devices that we use can take any number of forms. And I think that this is certainly a problem to be treated in research, is how to identify the types of forms that let you convey a great deal of information and look at interactions between a wide variety of variables at one time.

This is some more work that was done at the supercomputing center at the University of Illinois by Bob Haber. And here, we're looking at crack propagation. And I believe the height field is the axial stress in the material and the color has something to do with the sheer stress.

As a last example, this is again something that was done in the University of Illinois, and it's a very difficult simulation in which an awful lot of information is displayed. Here we're going to look at an injection molding problem and we're going to look at the flow of plastic into a mold. The simulation was done by Rich Ellison and Ray Eizak and really, the visualization technique, or the consultant for visualization was Donna Cox. And here we have the collaboration of scientists to actually create the simulation, and an artist to deal with the problem of visualizing and displaying information.

We have an iconic representation here that describes plastic as it's moving into the mold. There is a directional component by the orientation of these arrows. The length of the arrows describe the velocity of the material as it comes in. There is temperature information that's displayed vertically on each of the little icons. There is pressure information that's displayed on the base of the icon grid as plastic moves in. And we can see the entire process of this mold filling and we can look at the interrelationships between very many different elements, and as it fills, we can see what's going on.

I have a video of this that I'll show in just a second, but to sum up, we have multi-variate applications, and as manufacturers of software, certainly we would like to provide good software for you to do visualization. But the problem is not one that we understand, and in most cases, it's not one that you as scientists understand how to deal with this incredible wealth of data that's available and display it in a way that lets you see all the relationships you want to see at one time.

So I think what we can supply for you now is probably only a prototyping tool for those types of future applications. And we're waiting for you to show us how to build better tools that are going to let you look at this multi-variate data in meaningful ways.

So if we could roll the tape real quickly. This is the tape that was generated showing this simulation in progress, and we see the plastic moving into the mold. We see the fill patterns, the pressures, temperatures. And this was very useful for mold design and better understanding the things that are going on.

But it's difficult to do this looking at single slices, looking at only a pressure slice, or only a temperature slice, or only a velocity slice through the data. And we really need to see all the variables at one time. And eventually, we see plastic fill the entire mold. As it fills the mold, eventually, the temperature starts to drop, the pressure goes up, and we've reached the final stage for the part.

We'll just take one more look at it and if we could cut the videotape after this. And I'd like to thank all the people at the University of Illinois that supplied me with slides so that I could show this marvelous work. Thank you very much.

GORDON BANCROFT: At this point, we'd like to open things up to the audience. I had some canned questions, but I think we're a little bit short on time, so we'll just go ahead with audience questions.

Before we do that, they've asked us to tell you they don't want anybody up on the stage after the panel, so they can reset it for the next one. So if there are more questions for any of the panel members,we'll be in Room 303 immediately following this panel.

So, I guess we'll at this point start with -- how about the person in the back?

Q: Ingrid Carlbom, independent. I would like to comment on what Alvy Ray Smith said. You said that now that we have volume data and we don't have to worry about geometry, and I would like to disagree with that. It's true that if you only want a visualized data, your original data, yes, that's true. But in the medical field, for example, what you then want to do is ask questions about this data, what is the size of the tumor, what is the volume of the ... space in order to determine when to replace a hip, for example.

In the seismic area, you would like to ask questions like what is the volume of oil in place or what is the contact area and so on. And now you have to worry about geometry and those geometry problems are probably every bit as difficult as those we are familiar with.

A: ALVY RAY SMITH: Yes. In fact, you're absolutely correct. I failed to make one of my points, which was that there are lots of problems for which geometry is absolutely the correct way to visualize a solution or possible solution or to provide an intuition about the problem. No question about it.

I just want to point out that there are numerous large data sets for which the finding of a geometrical abstraction in the data actually adds artifact to the data rather than clarifying what the data has to offer. For those, then the direct imaging approach is the correct one. They're not in competition with each other. One of them is more applicable than the other.

BANCROFT: Could we have everyone give their name and affiliation before their question, please? Over here.

Q: Hi. Gene Miya from the Ames Research Center. I don't want to make too gross a generalization, but it seems that the panel in large part is telling us what we're going to be getting as opposed to asking us largely what we're going to be needing. And this is one of those tail wagging the dogs and solutions looking for problems kind of things.

I suspect that the reason why this is, is because as people learn about computing, they tend to lose empirical ways of doing things. Alvy gave a good case, which was figuring out -- putting a laser through smoke puffs. But we have to learn how to take empirical data in various fields.

Rob Cook, for instance, is a good friend, points out that he has to occasionally go outside and look at scenes, rather than spend too much time with the keyboard. And to reiterate what this woman behind is talking about, I think one of the most important software directions that we can go to for visualization -- it's not enough, it's a good start -- is that we need to be able to take numbers, literally, because that's how you make predictions, to how you come up with mathematical models and like, you have to take lengths.

You look at an image, that's fine. But when you talk ... God, that's nice, but how long is this distance? Or a harder one is, what's this angle ... maybe in a bond? What kinds of surface areas are we talking about? What kinds of volumes? And I don't see that in any of these tapes, slides, or what other -- because in large part, our users don't know. I mean, I don't know myself what I'm going to get sometimes when I take a look at data.

So you need also to provide, in addition to numbers, you need to provide points of reference. One of the things that makes some of Jim Blinn's videos so important is, he has the graph paper in the background, you even get some idea of what kinds of distances and ratios and proportions that are involved in some of these things, the scale.

It's a new field, admittedly. But there have been some computer companies who have been at this for a little while, like itty-bitty machines, as an example. I guess what you guys are going have to do is, you're going to have to bring in more scientists to talk to you guys -- I know that Mike Mailin (?) was at Pixar for a while -- and get a feel for why they're doing, as opposed to just telling us, well, here's the next generation of workstations. Mike.

A: MIKE KAPLAN: I think we are trying to address that, and we're not trying to address that by knowing what you want to do, because I don't know what you want to do. What we're trying to do is to make a framework that gives you a lot of functionality and utility and let you add things to it that are appropriate for your application. And the idea is to give you that framework that has a lot of functionality that you wouldn't want to have to program, like making real time displays work and handling attribute management and hierarchy and data base, and then say, look, here's an easy way to plug in, for example, an object that when it's displaying itself,

also displays an axis and a label, but only when it's between a certain set of angles, and when it's beyond those angles, it shows itself.

There are a lot of ideas that people have come up with for adding things like that, and we're trying to produce the extensibility in a standard way, rather than have everyone have to rewrite all the utilities themselves. So I think we're trying to address that.

A: AL LOPEZ: I'd like to agree with what Mike just said. In fact, I think that one of the values of an object orientated approach is that ability to dynamically extend the environment through user-written capabilities.

I'd like to add also that scientific visualization is really, for workstation manufacturers in particular, an emerging market. If you look back about eight years ago when workstations first appeared on the scene, ECAD type of applications were running on mainframes and minicomputers and when the workstation brought together networking, graphics and microprocessor compute power, memory prices dropped, it enabled those applications to move onto a single user network platform, the workstation.

Now ECAD applications over the last eight years have moved down through the mid range of workstations to the low end and actually fallen off the end and are now on PCs as well. And it was over that period of time that we learned more about the requirements of ECAD system developers and they learned better also how to make those applications work in a workstation environment.

The same thing is happening to MCAD when the graphics on the workstation became capable of supporting MCAD applications. And now we see scientific visualization applications that have been typically running on a variety of systems, including supercomputer architectures because of enabling technologies that have been brought into the workstation by systems like the Apollo VN10000, the Ardent machine, the Stellar machine, and even the Silicon Graphics system. Those applications have now been enabled to move onto a workstation platform.

We tend to focus a lot on where most of our volume is and as the volume increases, as those applications move down through our line, we'll be working closer with scientific visualization applications with scientists together, learning how to make those applications work and what requirements they really have. So I expect a lot more attention to becoming in this area.

BANCROFT: Okay, let's see if we can keep the questions as short as possible so everybody can get a chance. Over here.

Q: I'm Rob Wolff from Apple, and this is going to be a real question. I'm working on this, I'm taking an English course and all that stuff. The question is directed towards Mike Kaplan and Roy Hall. As a scientist, I really appreciate the functionality that exists with software like Wavefront and Dore.

But I don't want to have to take a training class to use it. User interface is critically important to scientists in order for them to be able to do science without having to go somewhere, take a training class, without having to know C or know UNIX in order to do these things. And without the user interface, you essentially don't have the access to all these marvelous tools.

So the question is, what -- I'm learning. The question is, what work and what efforts do you see in your respective areas are being directed towards this? That's a good question.

A: ROY HALL: Well, I think that we all have to assume that at some point we're going to build up our literacy in different areas of graphics, the same way we had to built up our literacy

in different areas of programing and perhaps had to learn to use C, or at least we had to learn to use somebody who knew how to use C, and perhaps we need to find somebody who knows how to use graphics to reduce the problem.

I think a great deal of efforts that we see, and certainly in the evolution of product that we've seen on the show floor over the last two or three years, that the user interface is becoming increasingly easy to deal with. Certainly in commercial animation systems, the needs of the commercial animator are substantially different than the needs of somebody doing scientific visualization.

And as the people who are doing special effects and those things need more and more effects, the menuing seems to get increasingly more complicated. And perhaps we need to branch off a simpler version for the applications and visualization. And I think that we're going to see those in the future. I think we'll see much easier user interfaces, things going to a more Mac-like form that 99% of us are familiar with and we can look forward to that in the next few years.

A: MIKE KAPLAN: Well, I think it's a very good question. And if I can ever get out of the trenches of making this stuff run on everything, that's my next project, or one I'm looking at very seriously. I believe I have some tools I can work with now that let me make systems that are quite flexible. And I believe other people using similar tools -- not necessarily ours -- will have those applications coming, and not just applications, but systems you can interact with and easily manipulate and visually manipulate data from analysis and steer analysis and do things with 3-D and 2-D on the fly. But first, there's an enabling set of tools, and that's what, at least we've been working, both hardware and software.

I think that other languages, like Smalltalk, may play a part in that area and I certainly think that the user interface management systems will play a part in that as well, a networked user interface and computation.

BANCROFT: Over here.

Q: My question is addressed to the panel as a whole. And all the panel members seeming to be addressing the visualization--

BANCROFT: Could you give us your name and --

Q: Oh, I'm sorry. I'm Gerald Edgar of Boeing in Seattle. And all the panel members addressed visualization of a single model. That is, computational fluid dynamics, thermodynamics of plastic injection and the like. And none of you have addressed being able to integrate and visualize information from various sources such as if you're able to visualize CFD with the stresses imposed by whatever source on whatever object you're modeling.

In the ViSC (Visualization in Scientific Computing) report that came out last year, there were two lines saying, "Graphics people aren't very good at data base, so we're not going to address that." Is there any future directions in this regard that you're aware of to address this problem?

A: ALVY RAY SMITH: Let me start. I think there's a great deal of interest in integrating data sources, the geometric with image sources together into a single visualization. But we're just starting and most people don't know how to do it yet. In fact, what the visualization report from NSF was all about, was to encourage research in exactly in these issues.

One small step towards an example solution is in the medical diagnostic imaging domain where several different modalities, as they are called, of data, are to be mixed together. For example, CAT scanners and magnetic resonance scanners are two very common types of volume

imaging devices in medicine and there are places, research places, that are combining images from those two modalities into a single visualization very profitably. Profitably in the sense that the referring physician gets a better sense of how to deal with his patient than he had with the two modalities separate.

A: MIKE KAPLAN: I've seen examples, beginning examples of this in the mechanical CAD field. Initially, people just did geometry and they did analysis. They showed the analysis as a table of numbers, and then they got the idea of attaching the numbers to the geometry with colors, and so on, and motion, and hence introduced at least two modalities into that.

But I've also seen people in the seismic and mechanical CAD field going further than that and showing two or three different effects on parts and so on at the same time. And I would hope that that sort of work will go a lot further, now that the technologies are improved, to make that real time and to make it easier to produce.

BANCROFT: Over here.

Q: Art Olson, Molecular Biology, Squibbs Clinic. The panel didn't really make very much distinction between what I would call working graphics and presentation graphics, in terms of the amount of effort that goes into the scientist's working day -- most of it is working graphics -- and also the amount of effort going into the software development.

There has been very little emphasis, for instance, on the question of tradeoff between rendering and dynamic interactivity, and also very little emphasis -- you don't see vectors anymore because everybody is working with surfaces. If you ask a scientist what he wants, rendering versus interactivity, at least in my field, interactivity is much more desirable.

I would like to phrase it as a question. What type of work is going on in trying to get whatever types of rendering, whether it be vectors and dots and not just surfaces or volume rendering per se, like we saw, just to go faster ...

A: None of us want to go faster, do we?

A: I think that a lot of the developments in speedups that go into the hardware architecture is to accelerate even surfaces. Also accelerate the rendering of vectors and the transforms that are necessary to redisplay even vector type or wire frame type of renderings or dot type of renderings.

In terms of software, I think that one concept that relates to this is the notion of demotion, which is that you'd like to be able to keep the rendering separated really from the data base or the geometry that you're using with which to render and be able to get a display of that information at varying levels of complexity, everything from the wire frame up through a ray trace image or a shaded rendering.

And I think that the interactivity as a function of how much hardware or the speed or the performance of the system that you run, what you'd like to be able to do is really turn a dial on your machine and pick the point at which it's interactive enough to satisfy your needs. And if you're on a machine that can do that with Gouraud shaded images, and if that meets the needs of your applications, then you can. If not, you turn the dial down and demote. And that concept really has to be built into the software systems that do the rendering

BANCROFT: Okay, over here.

Q: Ollie Jones, Apollo Computer. You know, scientists are suspicious people and there have been a lot of suspicious things about some of the images we've seen today, and I want to address the question to Roy Hall and maybe even to the chair as to what you think in the future the responsibility of the provider of visualization software will be to provide things like calibration bars to show what each color means and calibration to show whether you're distorting vertically and this sort of thing, and to what extent that's going to be left to the individual scientist, which of course is the ultimate responsibility.

A: ROY HALL: I think that we can address that issue pretty simply. And it relates to some comments that were made earlier about the tail wagging the dog, that very often, you as scientists are saddled with software that was built for the entertainment industry.

Well, the entertainment industry really pioneered a lot of the research that generated those techniques and that imagery, and many of the software companies, many of the people who are doing that kind of work and building that software, came from that environment. And I think it's perhaps naive of you to expect us to know what your needs are until you've done sufficient research to describe to us what it is you really need.

So I think until you know what it is you need, we sure as hell aren't going to know what it is. So it's really up to you to do your basic research to describe to us what the solution to your problem is, or to somehow get more involved with the manufacturers so that you can indeed tell us that we're solving your problem or not solving your problem.

Q: How are people going to do that? How do you suggest that scientists do that?

A: Well, I think we're seeing a great deal of it now at the University of Illinois. We're working with them pretty closely and they're demonstrating to us a lot of techniques that we never would have thought of because the problems we deal with were very much different.

And again, as I mentioned earlier, what we can provide right now is a prototyping tool. And the tools are there for you to make pictures that show those things and then to say, you know, we need to do that automatically, rather than to have to go through a lot of gyrations. Because in the end, it's our task to make visualization easy for you rather than to make it a new problem. But somewhere in between the solution of that task and our current state, somebody needs to do a lot of research to describe what the techniques are, because we don't know what they are.

BANCROFT: Good question. Thanks. Over here.

Q: I'm Bruce McCormick at Texas A&M, one of the authors of the ViSC report. I have a feeling that the scientific visualization report should probably have been entitled scientific visualization and modeling. And looking at Alvy's real data coming into there where you really want to be able to bottle it after you've got it for so many applications, I get the feeling that the panel is selling what they know, but they're not at all addressing the real problem.

What they don't know how to do is to do modeling off of that data. It's not computer vision. It's mostly volume imagery. I don't see any techniques down on the floor of the exhibit hall that show real modeling of real data. It's a very tough problem and I don't see any answers in any of the graphics firms that do it and I don't think they're even near it. And I'd like a little honestly about the fact that we've got a long, long way to do if we're going to be able to model visual data. It's a long way to go. Graphics is a lot of the answer and I agree entirely with the object oriented view, but I think you're selling what you know how to do, but you're not talking about the real problem.

A: MIKE KAPLAN: Well, it's very difficult to talk about things that you don't know about, and when you do, people really get on your case.

Q: Well, let's take some real cases, let's take the human body, take any of the CAT scan issue.

A: If you really do know about it, tell us.

Q: Well, we gave it to you in the visualization report. Take any of the human anatomy and try to do that. I mean, you can scan it in CT, in MRI, you can scan it histologically. You can use a laser microscope. You can do all these things to get volume data in. Try to model that stuff.

A: What you want to do is, you want to create volume models yourself. You want to model them as a CAD system would model a set of parts. But you want to do them volumetrically. Is that my understanding?

Q: Well, the data comes in to you volumetric, but you'd like to tear them apart, like they say in the old CAD system, you'd like to put retinas together that way.

A: There was a system called a Phoenix data system, Phoenix Data Systems that made a machine that did something very much like that and let you model voxil data and that company has now gone out of business.

Q: I talked with Maher just at the meeting here and has gone on in Paris, actually, since then. But that's irrelevant. What I'm hearing is, you know how to do graphics. You don't have a clue how to do the image analysis, and you're telling us that just a few more twists on the standard old graphics package and we're home free. And I just think that's hooey.

A: Come on, Bruce.

A: AL LOPEZ: I'd like to respond to that as well. I don't think that we can expect to be at a point where we have all the answers in the very beginning. And I really see scientific visualization, at least in terms of the workstation market, is a beginning in emerging technology.

Up until a year or so ago when we started developing supercomputer type capabilities into a workstation, we didn't even have the term scientific visualization on our slides as a market. The packages that are now being used in scientific visualization, like the Wavefront packages and so on, were regarded as packages for the entertainment industry up until only a couple of years ago.

I basically think that there's a lot of work we've got to do. There's a lot we both have to learn and we've got to grow up together, and it's just going to take time, years, and a lot of discussion, a lot of applications, working on these systems and a lot of requirements being placed on these systems in order to make applications work before we get to the point where we have those answers.

A: GORDON BANCROFT: I think we have to, at least in some sense, bite off as much as we can chew and solve problems that we can identify now and experiment with the solutions to those problems to go on and solve the bigger ones. And I agree that we've only just scratched the surface.

BANCROFT: Over here?

Q: My name is Bob Hendricks and I'm from the Johns Hopkins Applied Physics Laboratory. I work in video production at the laboratory and our group does video productions that generally

serve as status reports to the sponsor for projects we're working on and so forth. And I need a 3-D graphics animation system that will create quality images that I can use in the video post-production environment. But at the same time, since we are a scientific laboratory, I'd like the system to be able to accept technically correct parameters instead of just the kind of things that video production people are used to.

I'd like to ask directly the panel, going down the line and ask you what your opinion is on what the best software is that's available today to address both of those needs.

A: Gordon Bancroft: I think you'll get four different answers there.

A: I think the chair should answer that one.

A: GORDON BANCROFT: Ah, gee. Don't make me do this. I think in my estimation, there are a variety of different animation systems and ones you can build yourself and ones you can buy completely put together for you. My gut feeling is that there's probably a certain one that's best for each application. I guess if that gets me off the hook, I'll go onto the next question. Mike's going to tell you that it's Ardent. Just kidding.

A: MIKE KAPLAN: Not at all. I think my answer would be, I don't know of one that's really great, that you can just pick up and use. But I would say that the animation systems that we're seeing for the entertainment industry or from that are probably not nearly as applicable as the CAD systems that have been developed over the years. In other words, which are not primarily into animation, which are primarily into design and manufacturing. Those systems tend to add animation capabilities later and they don't tend to be as fancy as the capabilities that the video animation people have done. But they're much more concerned about tolerance and materials and showing the real stuff.

So I would say that with the combination of photo realistic rendering or high quality real time rendering to these CAD systems and some of the new CAD vendors that are coming up, may provide more of an answer than just an animation system.

BANCROFT: Over here.

Q: ... Geo Systems. When we look at the scientific process, what do we do, as a scientist, myself being a physicist? We start measuring data. We have measurements. You're providing visualization, and we talk about visualization here, but that's only one very small part of the whole scientific process.

The next thing we want to do is, we want to be able to extract numbers from those data which we are visualizing and then put them back into .models and perhaps revisualize them. What is being done about trying to actually grab measurements being provided in these computer data bases? If we don't address that issue, you're not addressing the scientist' needs.

A: ALVY RAY SMITH: Well, in medical -- I'll use medicine again as a place of quite active research today on extracting measurements from medical imagery. Just as you say, one of the most interesting things to, say, an oncologist, is the size of a tumor today on such a such a patient. So what they're doing is extracting volume data from their image data. I think that's going on right now in various research centers around the country.

Q: Do you see a generalized set of tools coming up for that?

A: I certainly do. I guess what I'm pointing out is that you're absolutely right. People want to extract measurements from their data, whether it be geometry based or image based. You can still extract measurements from it, and I expect to see package come along very shortly from the research that's being done currently. This stuff just started.

BANCROFT: I think we have time for one more question. We've got to wrap it up here. Over here.

Q: Charles ... from the University of Maryland. The computational data when you want to visualize, I think that naming the scientist is somewhat misleading. Usually, there are two scientists involved. One who has the original problem, the other who has to solve it, and sometimes they called ... mathematician.

In this case, using finite elements, the model for geometric space is even in the finite element form. But you apply and if visually shown otherwise, it's misleading. And whatever that applied mathematician's tolerance is, it would be completely wrong visually if you model it differently. And I feel that nowhere was mentioned that finite element -- it's very much connected to the finite element modeling when you might want to visualize it.

The other thing, maybe a long-range one, many of the problems are not spatial geometry but parametric geometry. I can parametrize shapes geometry, and eventually it would lead, on a long-range optimal programming, it might be five years later, and in that sense it could be very commercially available, especially if one of the parameters could be a dollar sign. I would like to have some comments from the panel in this regard.

A: Just to be honest, I missed your second question. But I think the first one had to do with, perhaps visualizations are misleading.

Q: The scientists, in respect when you are considering, the chair mentioned that, what ... you visualize, the model or the modeler? So there is a mathematical model which you have to solve. Originally, it was not a mathematical model, but it was originating scientist wants to know. Once it gets a mathematical model and solve it, then an applied mathematician comes in, because the solution for that mathematical model is again an approximation. So your data is not fully accurate. You are getting for visualization.

So there are two things which you can model here. Model something for the scientist or model how the solution phase goes and how accurately the solution goes in the scientific solution. Because finite elements are used. The finite element itself specifies, if for example, you represent surface, then --

A: We need a question here.

Q: I wonder if ever it was considered that one should contact a finite element analyst and look at the representations through him, not just by the originating scientist.

A: Is the question, have we called in finite element experts?

Q: Yes.

A: Yes. Well, I showed one example that was exactly based on finite element analysis. But I'm not sure that's the answer to your question.

Q: But I haven't heard that when used that type of element surface volume element, but the finite element defined for the solution.

BANCROFT: I'm afraid we've run out of time here. We've just gone over the time limit. Thank you very much for coming.

Scientific Visualization

Presenting complex, quantifiable physical phenomena that has been either mathematically modeled or scientifically measured so that it can be meaningfully and understandably displayed for analysis in 2, 3 or n dimensions.

Software Directions for Scientific Visualization

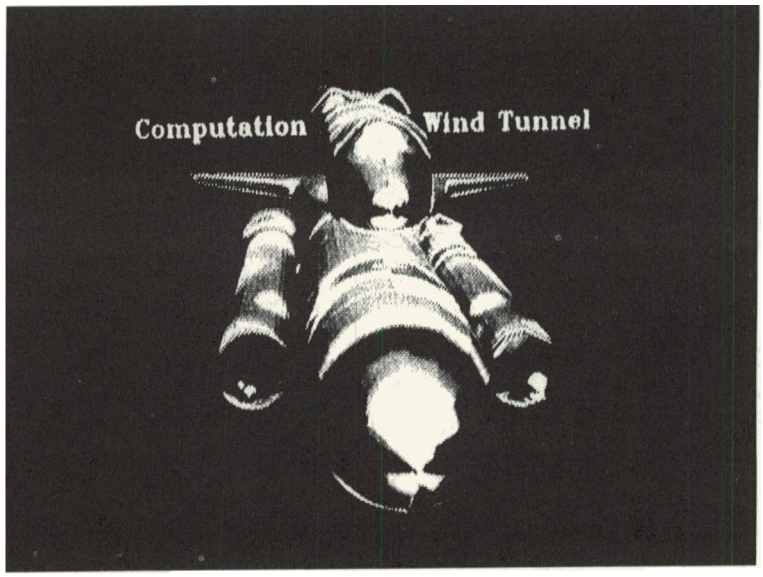
- Visualization standards
- Software standards and new architectures
- Modeling the environment vs. modeling the model
- Scientists are not after photorealism
- 3-D techniques and stereo

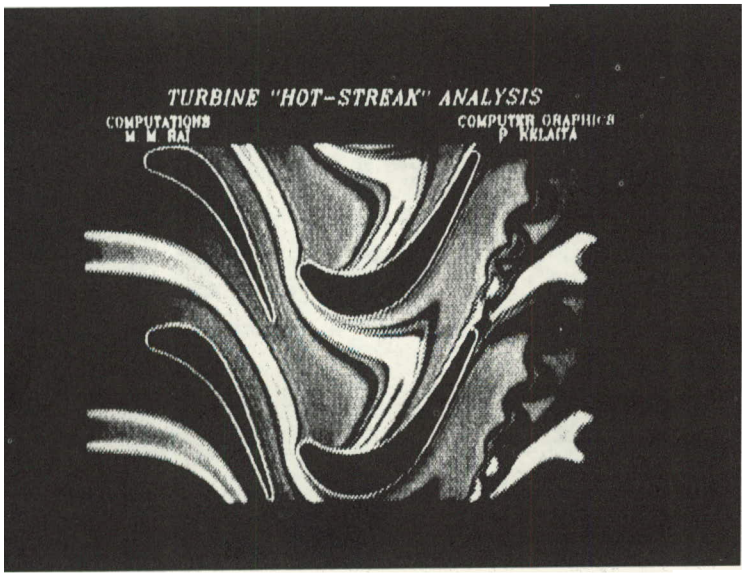
SIGGRAPH '88

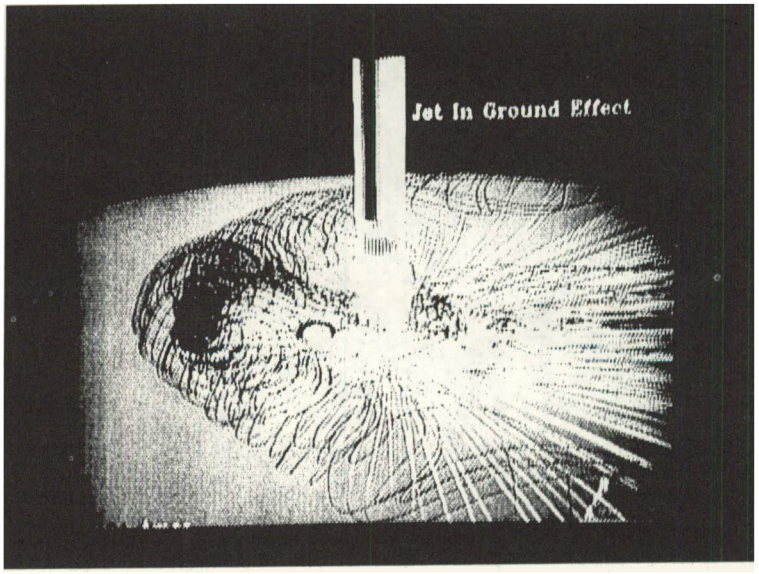
Software Directions for Scientific Visualization

- Volumetric Visualization
- "Immediate" mode drawing
- Distributed Graphics Processing
- Digital WKS to Digital video
- Unsteady Phenomena

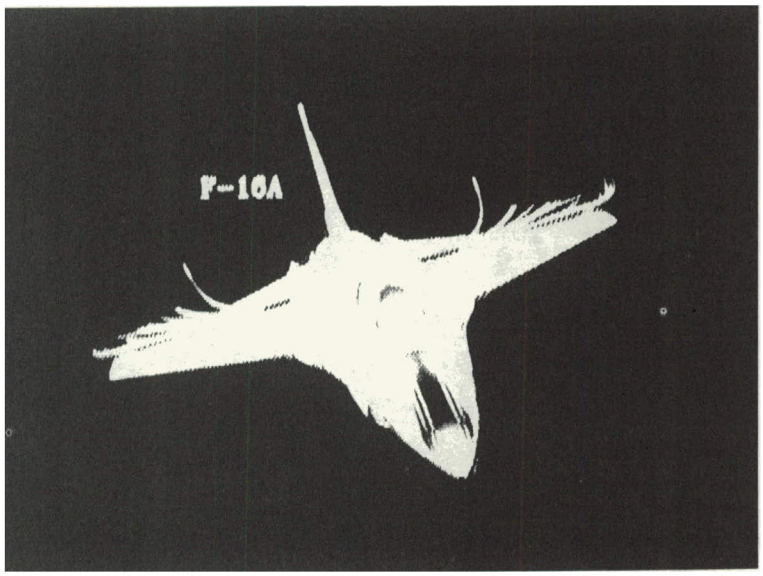
SIGGRAPH '88

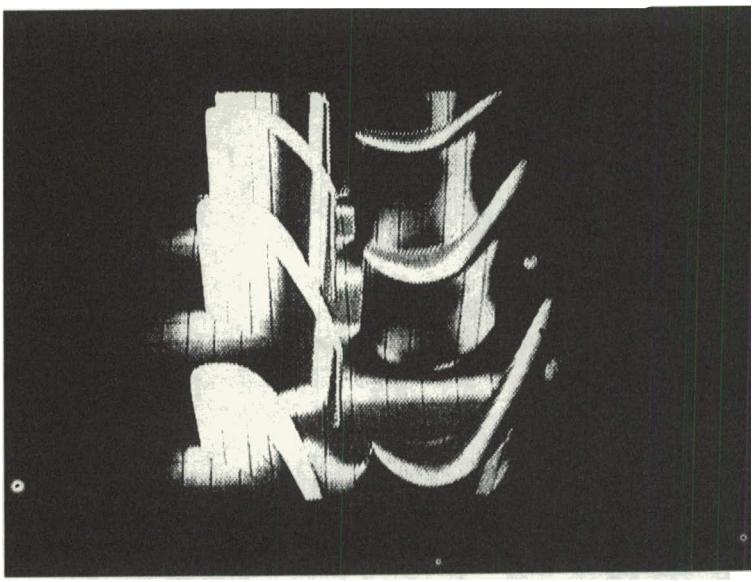




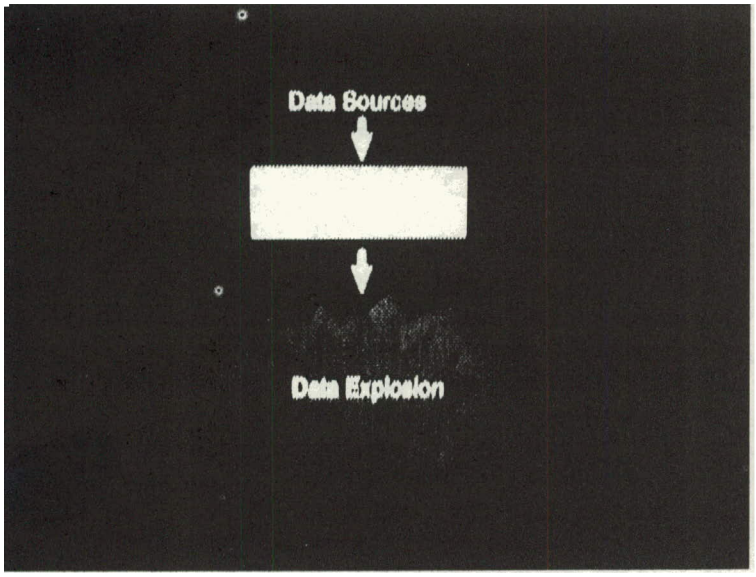


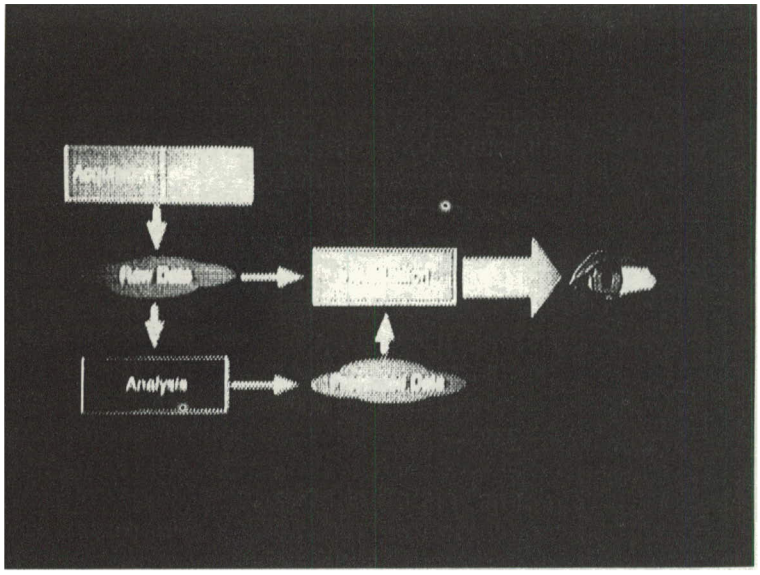
Gordon Bancroft

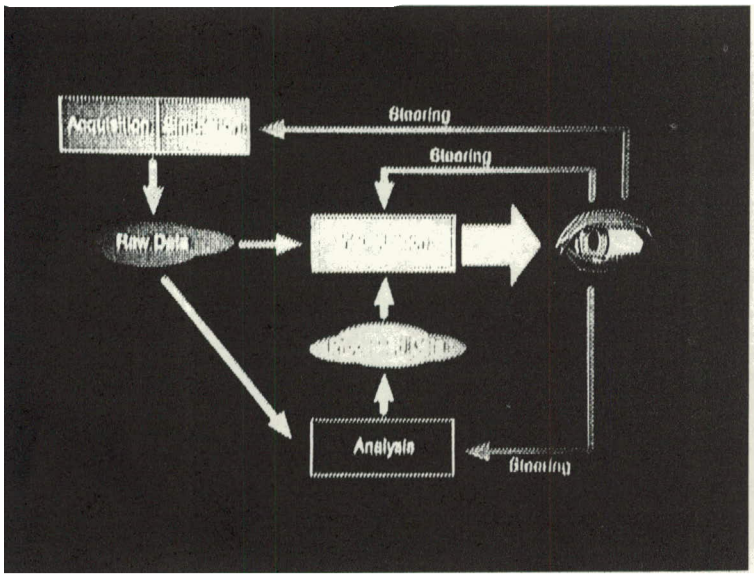


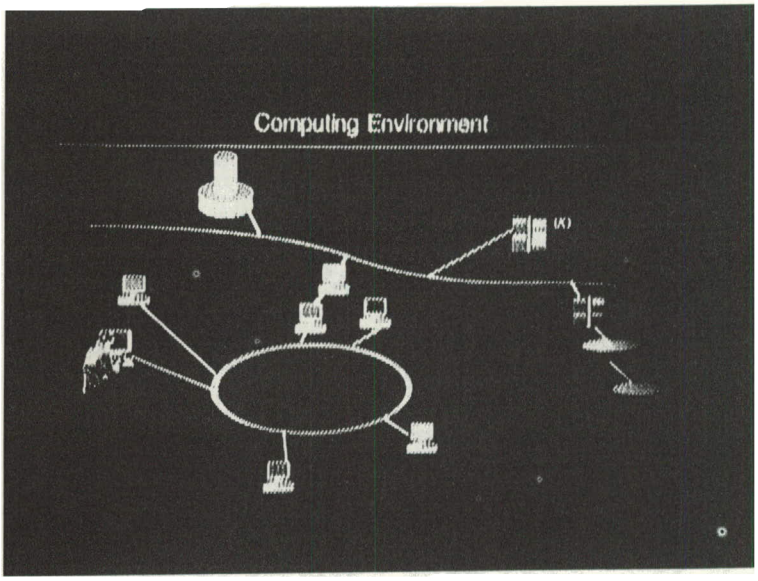


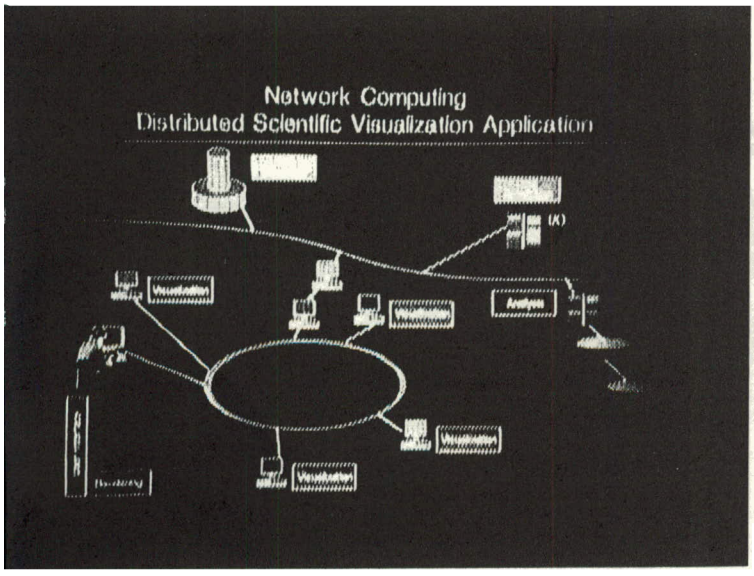
Gordon Bancrot

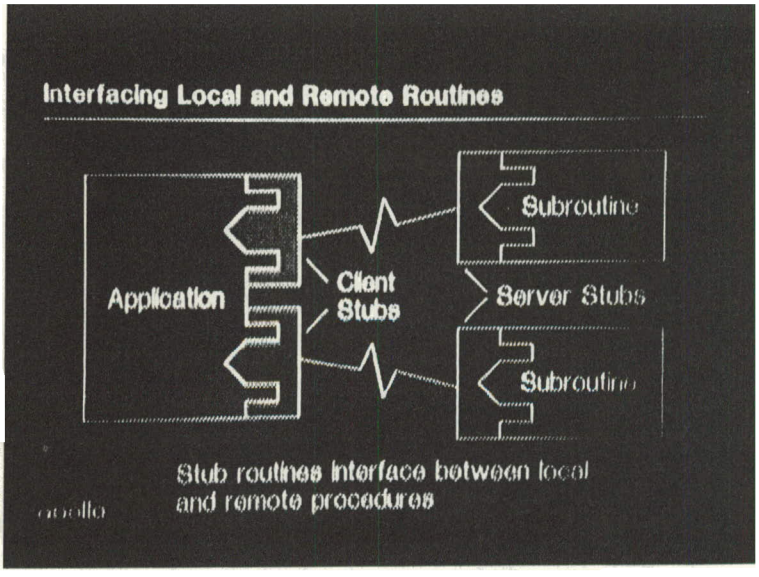

Data Sources
Data Explosion


Analysis


Steering
Steering
Raw Data
Analysis
Steering

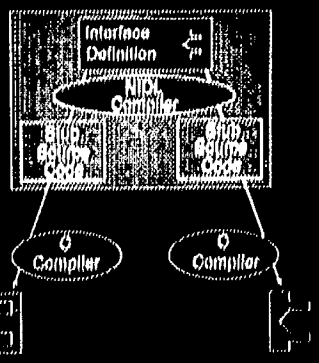

Computing Environment


Network Computing
Distributed Scientific Visualization Application


Interfacing Local and Remote Routines
Application
Client Stubs
Server Stubs
Subroutine
Subroutine
Stub routines interface between local and remote procedures
apollo

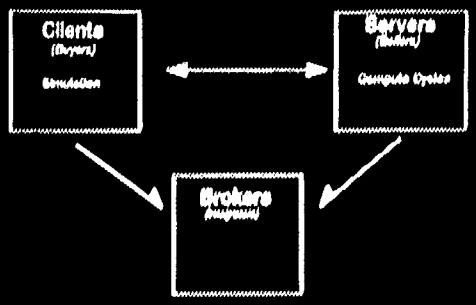
Al Lopez

Interface Definition Compiler Generates the Stubs



apollo

Client Server Broker Model



apollo

Trends and Directions

object oriented graphics system + object oriented network computing system = A framework for object oriented distributed graphics

Al Lopez

Network Computing in Scientific Visualization

Al Lopez, Director
Graphics and User Environment
Research and Development

apollo

Effects of Visualization

- Higher bandwidth to scientist

Effects of Visualization

- Higher bandwidth to scientist
- **Greater insight**

Effects of Visualization

- Higher bandwidth to scientist
- Greater insight
- **Desire for more interaction**

Effects of Visualization

- Higher bandwidth to scientist
- Greater insight
- Desire for more interaction
- **Demand for:**
  - **Higher bandwidth communications**
  - **Higher computing throughput**

Environment

Local and Wide Area Networks

- Hyperchannel 50Mb
- Ethernet 10Mb
- Token Ring 4, 16Mb
- Twisted Pair 10Mb

Heterogeneous Computing

- Supercomputers
- Mainframes/Minis
- Workstations/Superworkstations
- PCs

Attempts to Satisfy Desire for Higher Bandwidth

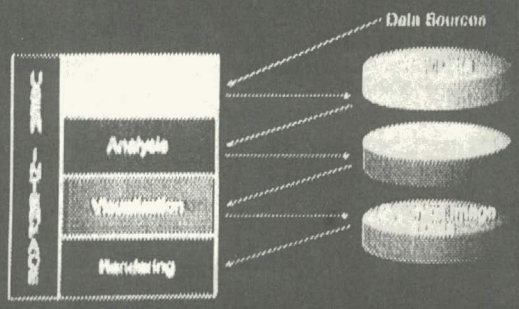
- Brute force approach
  - FDDI (100Mb)
  - Faster CPUs

Brute force alone is self defeating

Additional Solutions

- Architectural approach
  - Placement
  - Allocation
  - Distribution

Application Taxonomy



Network Computing

- Remote procedure call

Network Computing

- Remote procedure call
- Network Interface Definition Language (NIDL)

Network Computing

- Remote procedure call
- Network Interface Definition Language (NIDL)
- Object oriented approach

Al Lopez

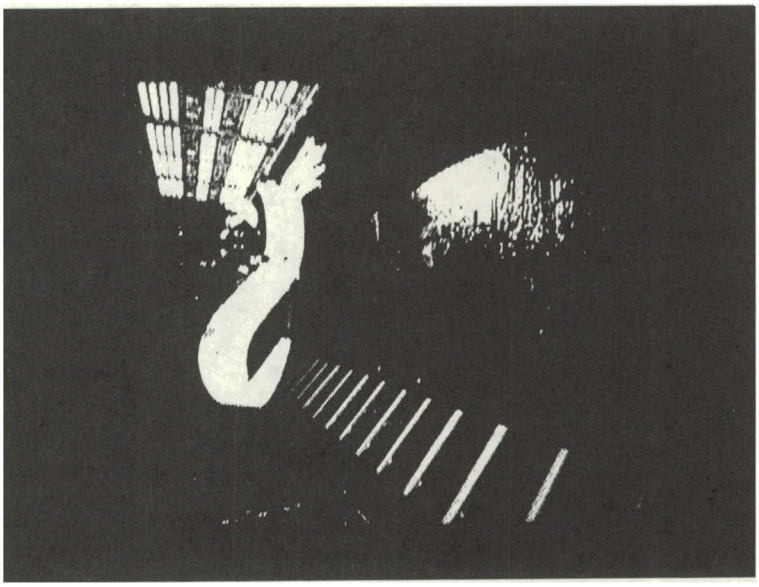
Network Computing

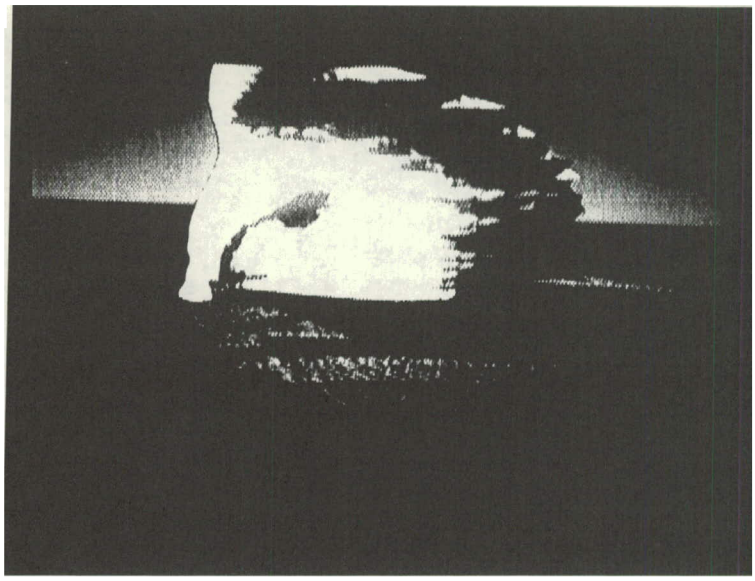
- Remote procedure call
- Network Interface Definition Language (NIDL)
- Object oriented approach
- **Location broker**

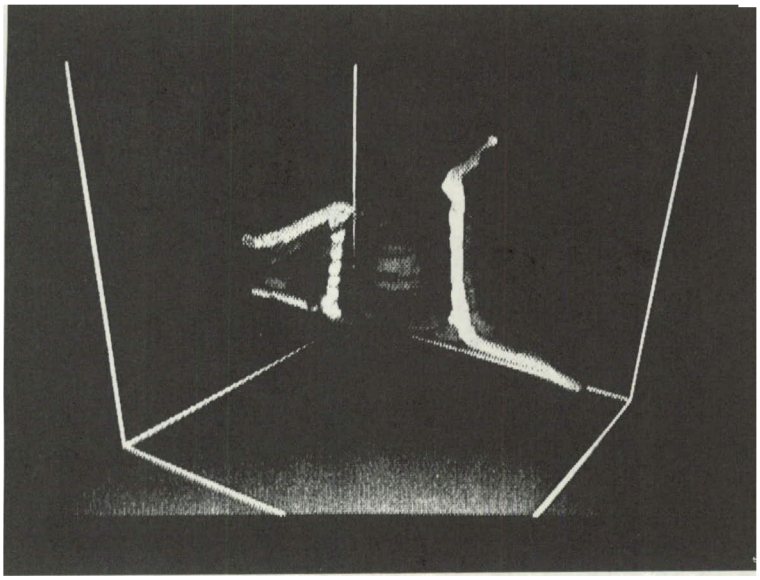
Trends and Directions

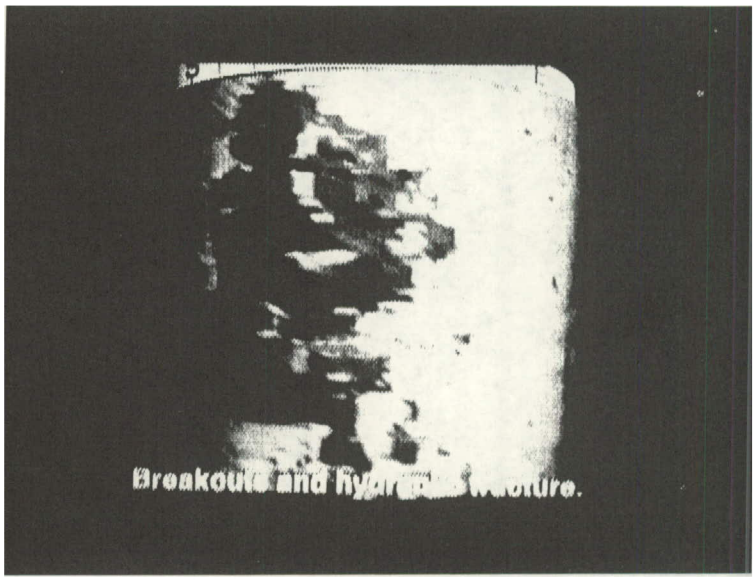
Object Oriented Distributed Graphics

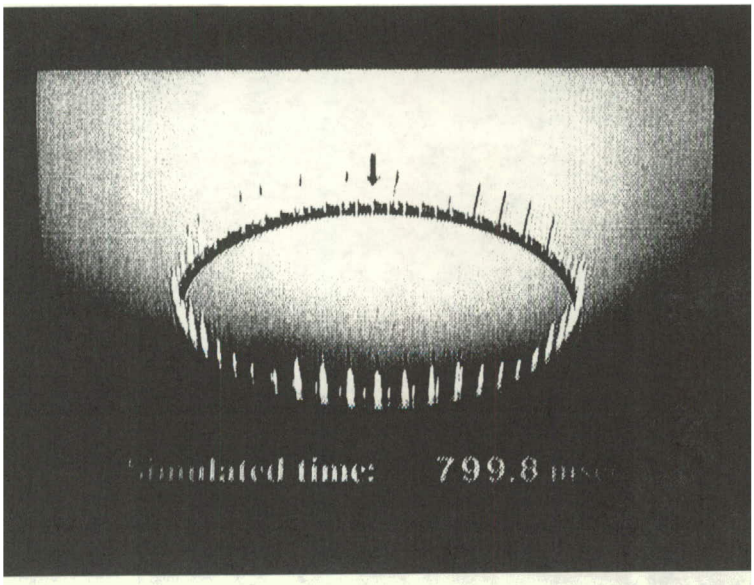
- Data abstraction
- Dynamic extensibility
- Remote objects
- Distributed computing objects
- Object programming model

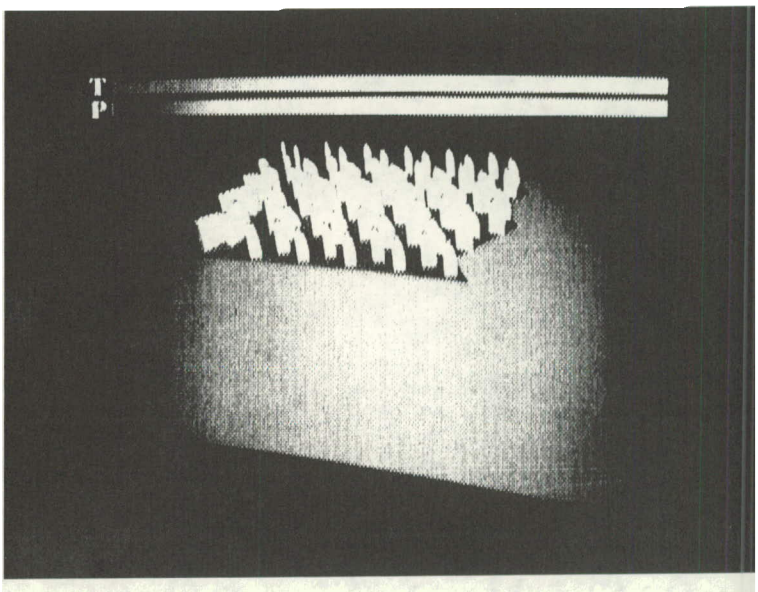




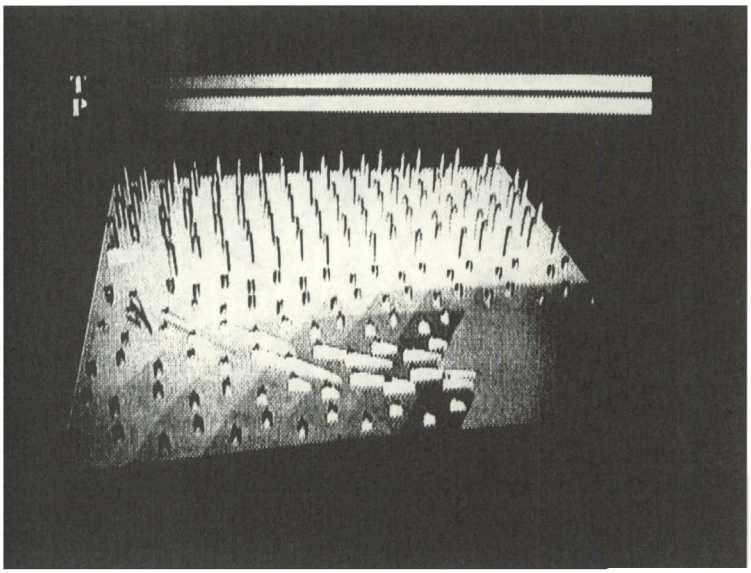




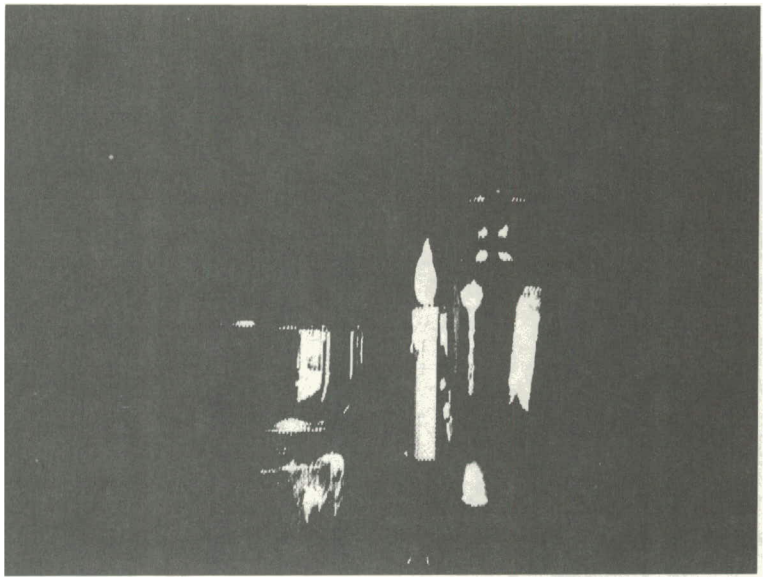


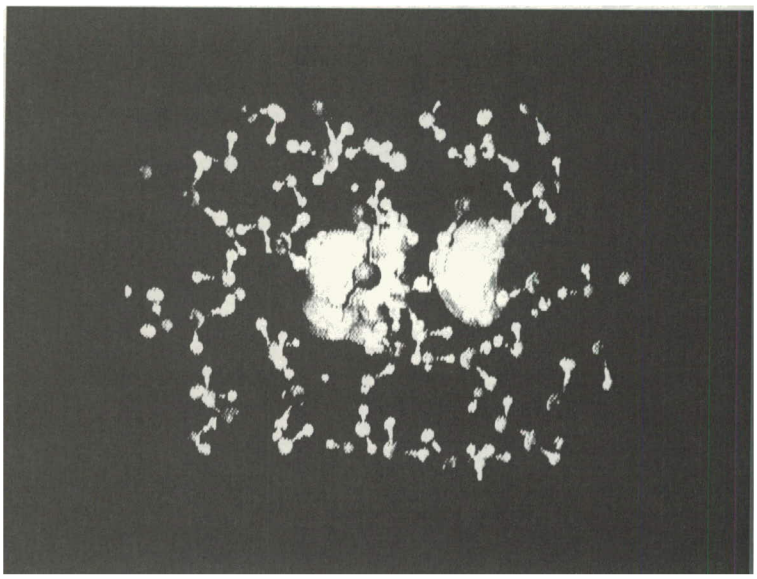


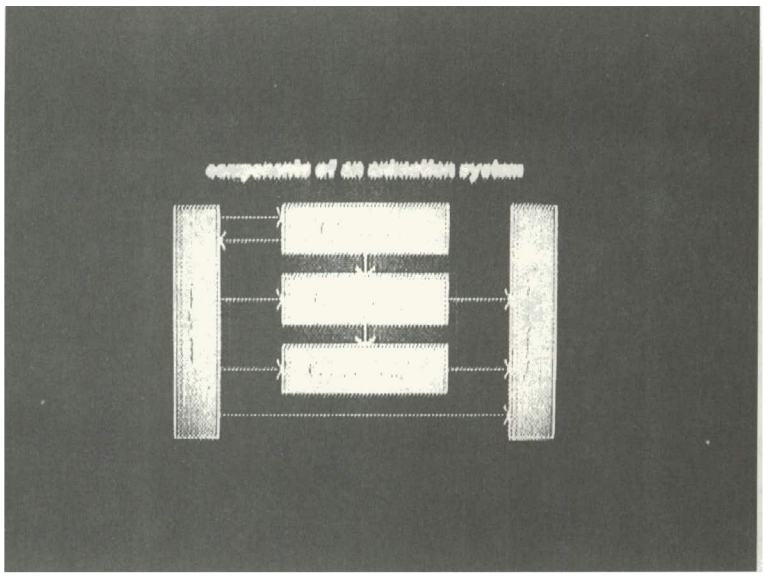
Roy Hall

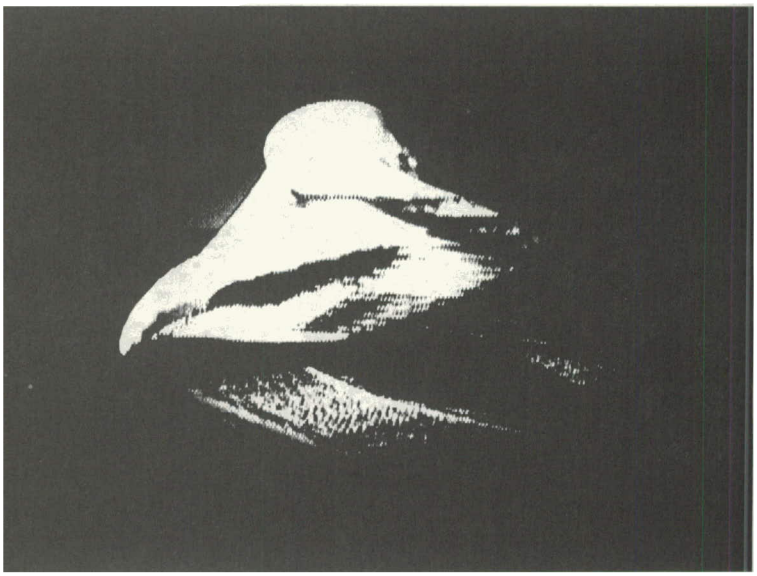


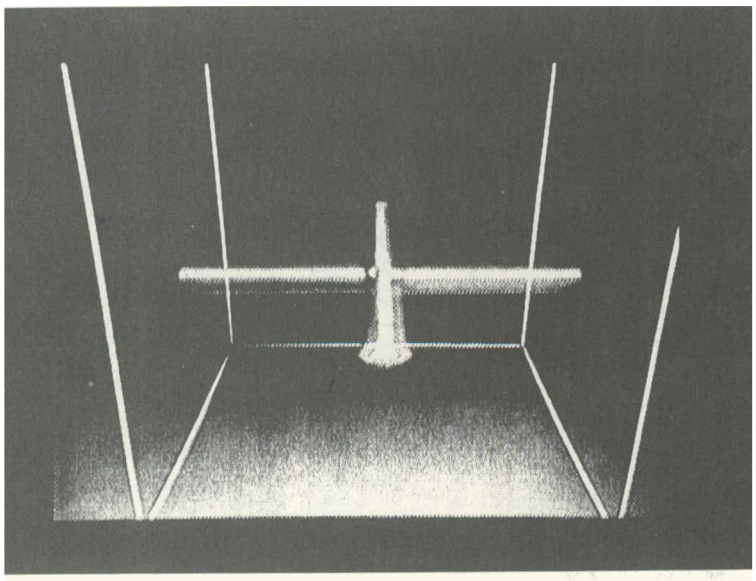
Roy Hall



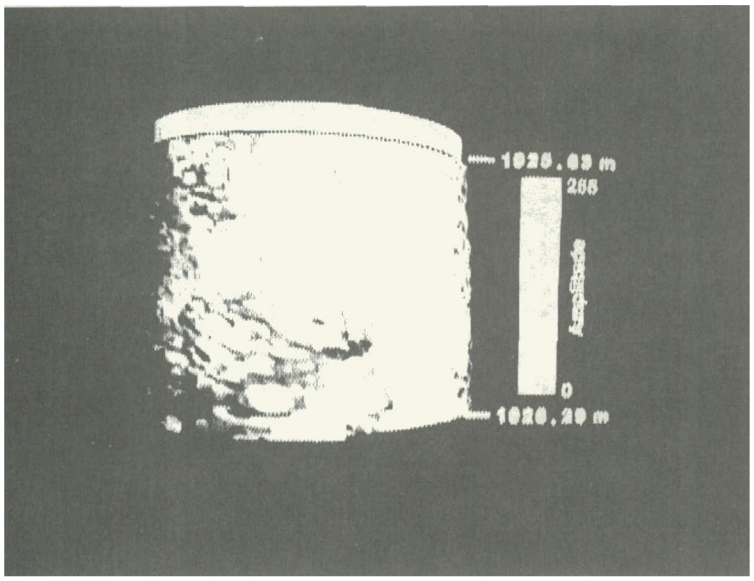


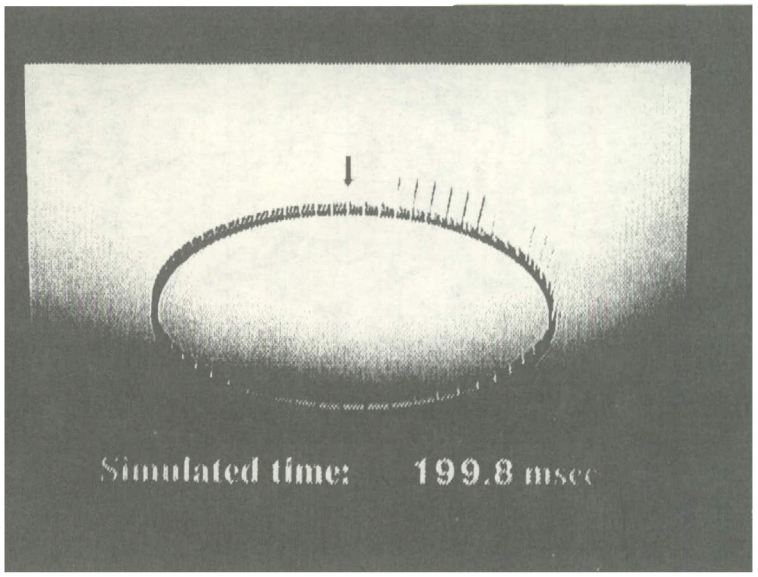


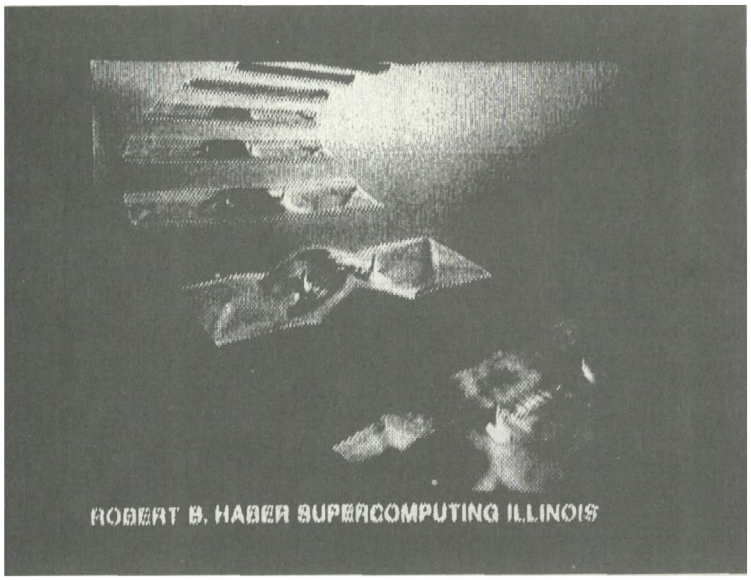


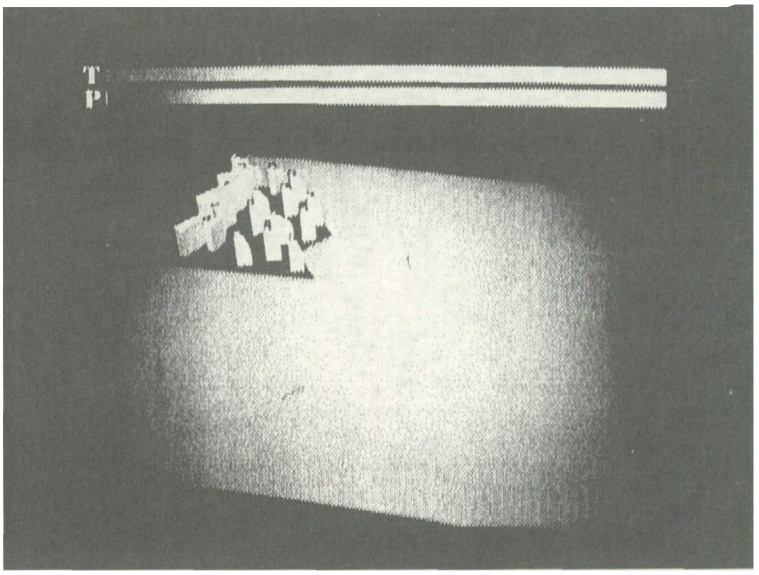


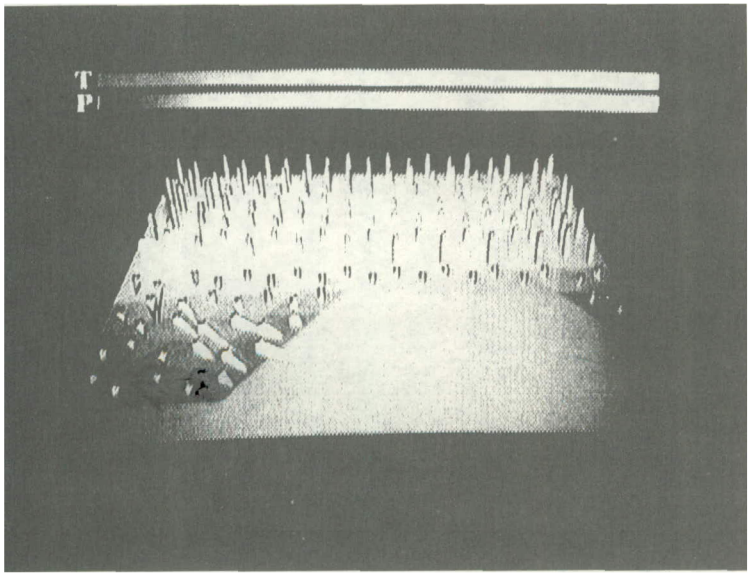
Roy Hall


1025.63 m
255
Amplitude
0
1020.20 m

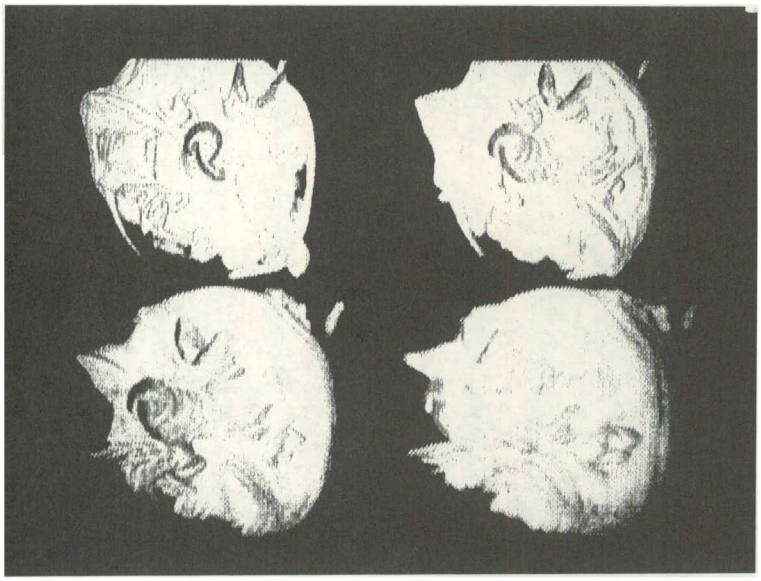

Simulated time: 199.8 msec

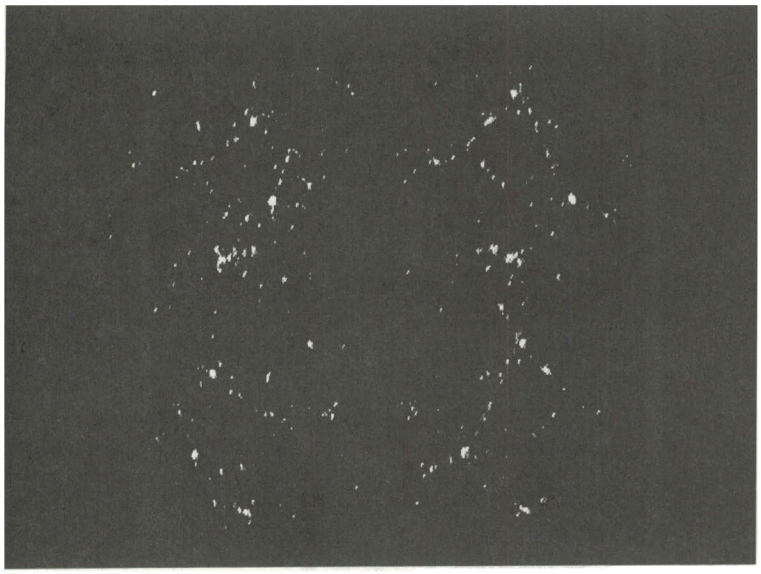

ROBERT B. HABER SUPERCOMPUTING ILLINOIS

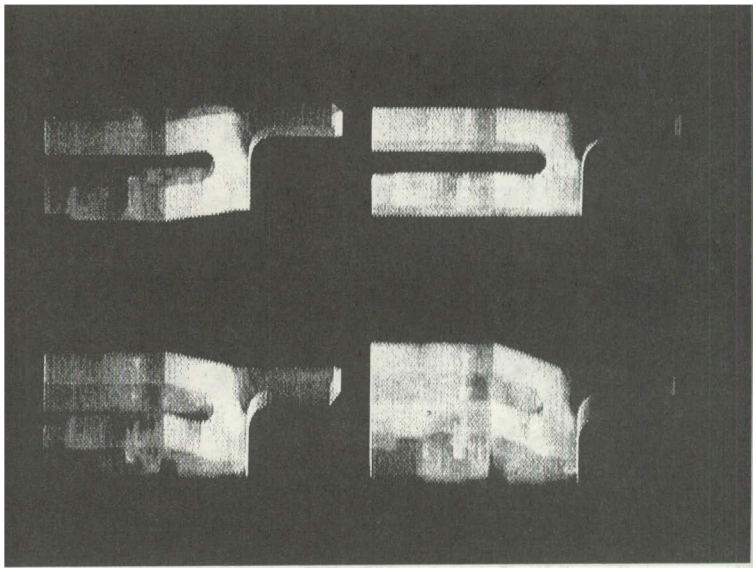


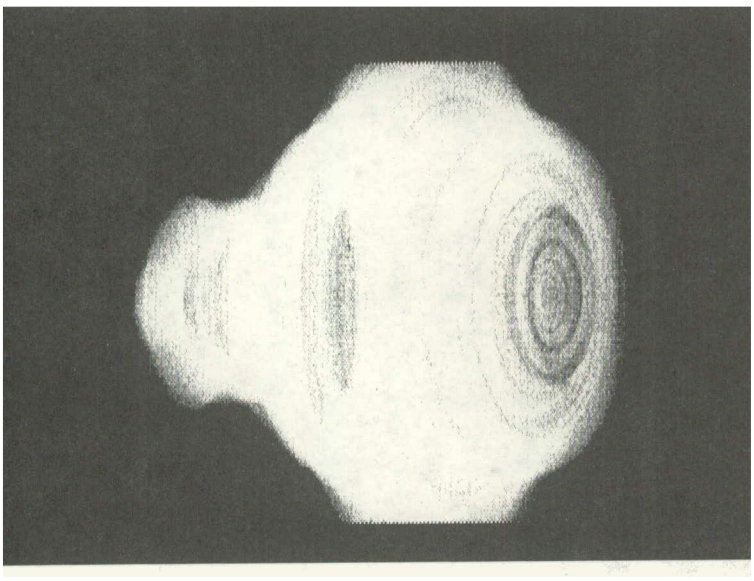


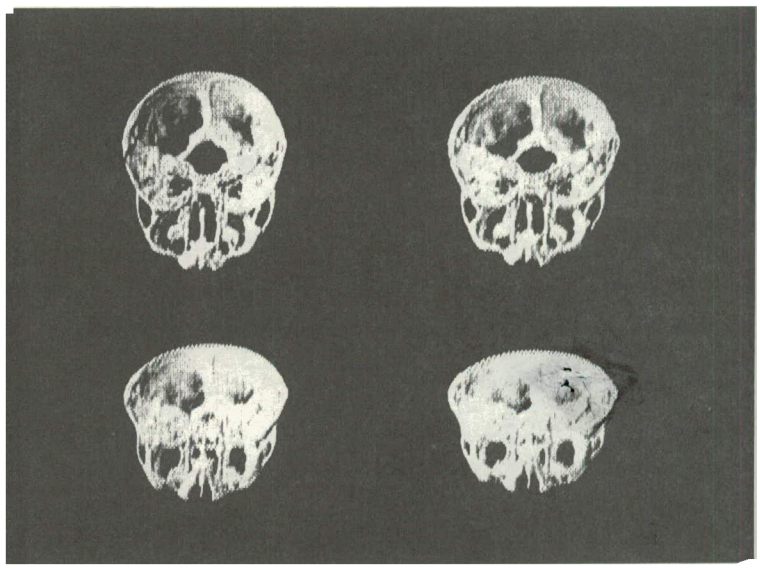
Roy Hall



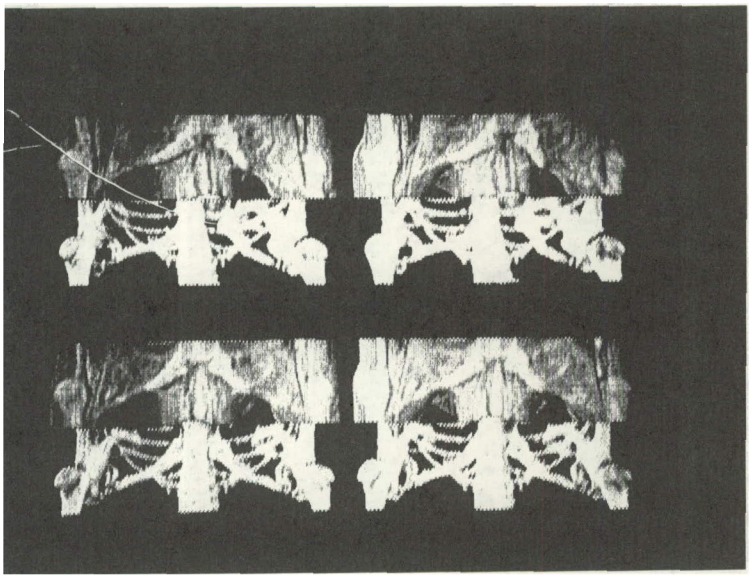


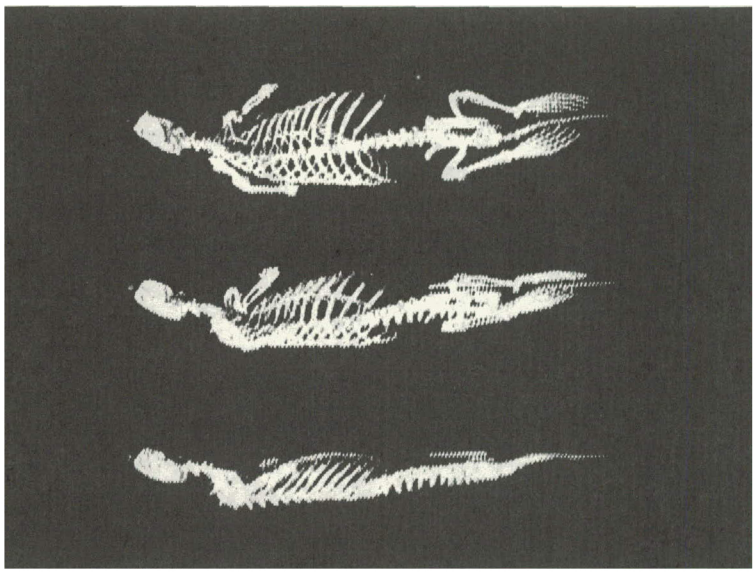


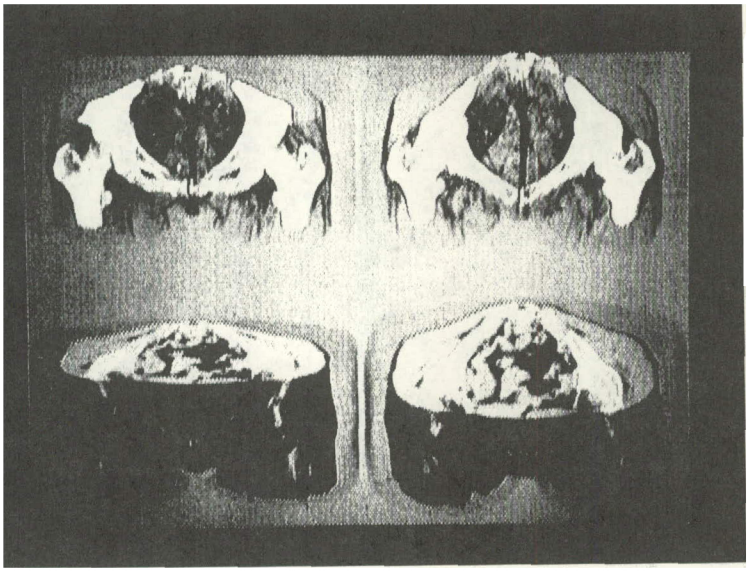


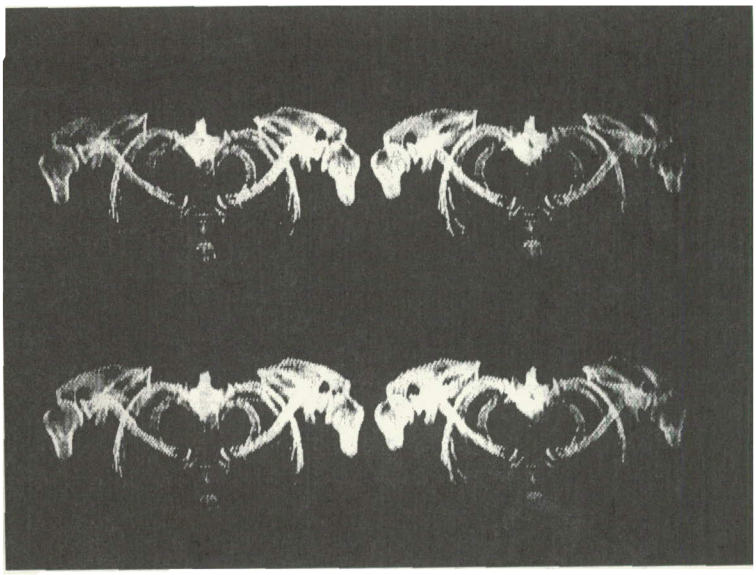


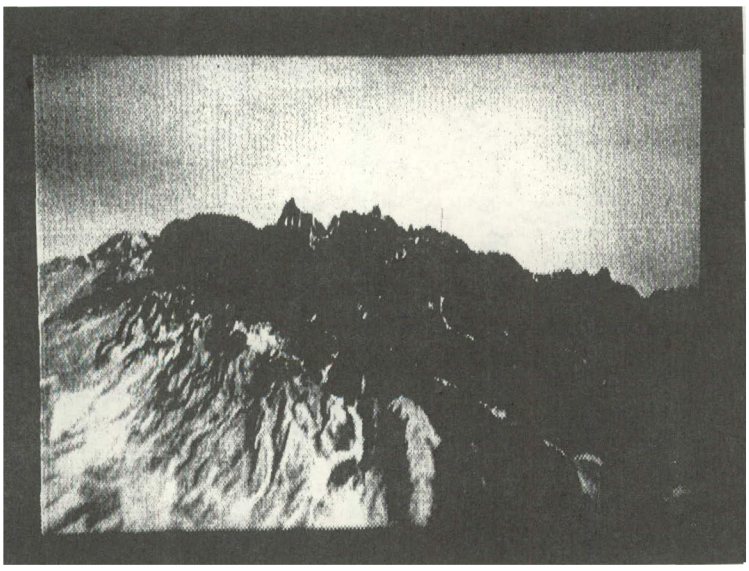
Alvy Ray Smith



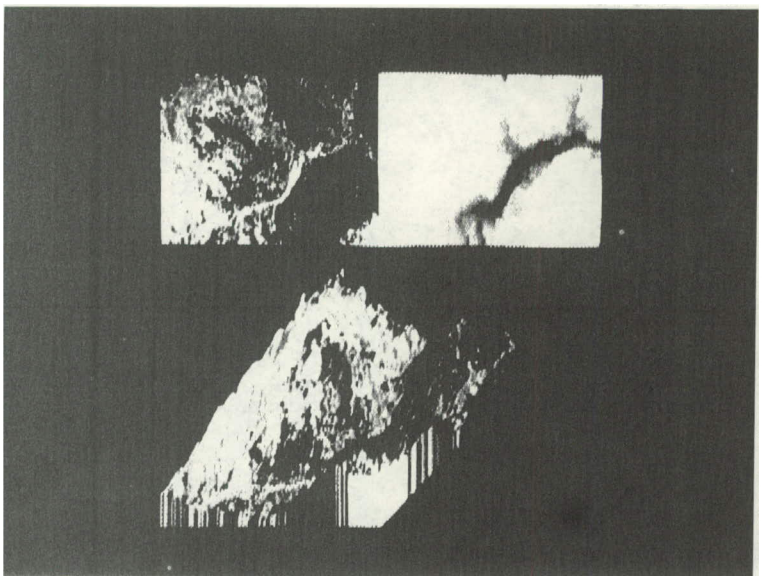


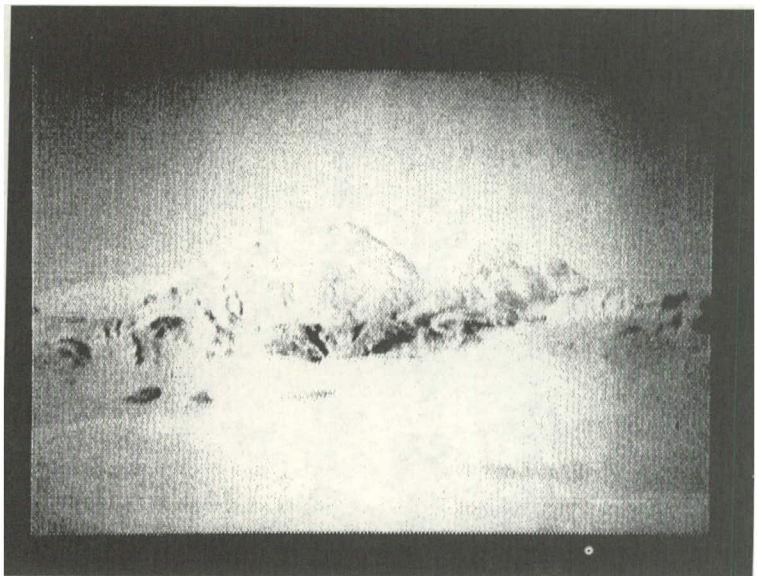






Alvy Ray Smith





Alvy Ray Smith