Collaborative 3D Visualization with CSpray

Alex Pang
University of California, Santa Cruz
Craig M. Wittenbrink
Hewlett-Packard Laboratories

Abstract

As networking technology brings us into the information age, 3D and multimedia applications become more prevalent and the emerging norm. Improvements in communication infrastructure also allow us to explore the realm of group interaction and collaboration over the information superhighway. Our work focuses on providing a small group of geographically distributed scientists the means of sharing their data and interactively creating visualizations and analyzing them. This allows for shorter turn around time compared to more traditional means which in turn allows for greater research productivity. The key features of our system include: different levels of information sharing, incremental updates to reduce network traffic, an intuitive floor control strategy for coordinating access to shared resources, a built in session manager to handle participants who either join late or leave early, and a host of collaborative 3D visualization aids. An added feature is that while the system is designed for small group collaborations, it can also be used for briefing a larger audience. These features are included in the CSpray *collaborative 3D visualization system.*

Key words and phrases: data sharing, scientific visualization, floor control, session management, shared virtual workspace, multimedia, ATM, computer supported collaborative work.

1 Introduction

Changing technology often brings about a change in how people do things. Two cases in point are the impact of massively parallel processing and the availability of large and inexpensive memory. Both have allowed scientists to look at larger problems and more realistic and complex models. But more fundamentally, they have, at the very least, influenced the way algorithms are designed as well as a shift in focus to more basic questions. Now imagine for an instant the availability of very fast networks – so fast that large amounts of data can be transferred almost instantaneously. Such technology would certainly change the way we do business. In some areas, it would in fact revolutionize how people interact. For example, instead of exchanging data by postal mail or even electronic transfer through email or ftp, geographically distributed scientists might collaborate by interactively visualizing and analyzing data sets on the fly simultaneously, thereby, reducing days or weeks of turnaround time to a few seconds. And just as other technological advancements might change the focus of research, this one might shift the focus from compression efforts to more human oriented interactions over electronic links. Even before networking technology matures to fully support such rapid data transfers, we can start to plan how such technological change might impact and change the way we do business.

One important crossroad of 3D graphics and multimedia over the information superhighway is the realization of collaborative 3D visualization environments. With a few exceptions, most visualization systems to date still operate in single-user mode. As with single-user tools, visualization products, or images, are created from one graphics workstation. Users may run remote modules, for example on a supercomputer, but they do not interact with other users in the creation of the visualization products. In contrast, extending single-user visualization tools into the *collaborative scientific visualization* settings allows multiple investigators to share data, views, manipulation sequences and to participate in the creation of the visualization products across the network.

There are many excellent examples of group collaboration dealing with networking issues to human interaction issues that are reported in books such as Groupware [1] and in proceedings such as Computer-Supported Cooperative Work (CSCW), Human factors in computing systems (CHI), and Graphics Interface. However, only a handful deal directly with issues associated with distributed and collaborative 3D visualization. We discuss some recent work in this focus area below.

NPSNET [2] uses IP multicast to manage very large distributed battle simulations over the Internet. This is achieved by partitioning the virtual battle field based on spatial and temporal proximity as well as functionally related entities. DEDICATED [3] is a distributed heterogeneous multimedia learning environment with a device-independent and application-independent architecture. Similarly, the SHASTRA architecture [4] provides an application-independent substrate as demonstrated by several scientific and engineering design applications. The HIGHEND visualization software [5] allows cooperative flow visualization of computational fluid dynamics (CFD) simulations. Likewise, Tempus Fugit [6] also provides shared visualizations of CFD simulations via calls to a distributed library (DLIB). A recent enhancement of a popular CFD software FAST [7] allows sharing of scripts and simulation data using a web browser interface. Efforts are also being made to make the collaborative experience more intuitive. For example, the LinkWinds [8] allows dynamic interconnection of multiple windows through a data-linking paradigm for a graphical spreadsheet look.

Designing a system for collaborative scientific visualization requires a new look at what it means to visualize data by groups. There are many new design issues, including the user interface, authorization levels on data and users' control. This paper reports how our collaborative 3D visualization system supports:

- Synchronized virtual workspaces for group generated visualizations;
- Private workspaces for visualizations generated independently of a collaboration group;
- Local data integrity, protection, and independence from the collaboration; and
- Several collaboration aids particular to remote 3D collaboration.

These features are included in the *CSpray* collaborative 3D visualization system. *CSpray* (pronounced sea-spray) stands for Collaborative Spray and is a collaborative version of *Spray* (see sidebar). *Spray* uses a spray painting metaphor for intuitive interaction with visualization tools. Spray cans are pointed and sprayed into the data set to create graphics primitives such as contour lines, streamlines, isosurfaces, etc. This is described in more detail in section 3. The paper will first present the architecture of *CSpray* in section 2, followed by details of the different collaboration aids provided by *CSpray* in section 3. In section 4, we describe how access to shared objects are managed. Finally, section 5 describes our experiences with *CSpray*.

2 CSpray

CSpray is architected for evaluation of ideas in resource allocation, and to provide an efficient and simple means for developing collaborative user interaction. We started with numerous ambitious goals in developing a testbed for evaluating resource allocation issues in collaborative visualization. A primary goal was the efficient exchange and communication of visualization information. The exchange of the information requires sending of the data sets, visualization products, or an intermediate form of a visualization product that can be rendered by the receiver.

Our communication design goals included defining the control protocols and mechanisms. Such control information is necessary to implement floor control, session management, robust handling of network/workstation failures, and intuitive handling of remote participant interactions.

Once the communication or protocol has been decided, the next goal to decide is how many participants can join in. This is a function of both the user interface, and the effective bandwidth resulting from the network/implementation. Our architectural goal was to have complete one-to-many connections to support full exchanges among all participants. We felt the scalability in this most general communication model, could be addressed by fast networks, efficient implementations, and multicasting protocols.

Because scientific visualization is but a window upon a world of computed, sampled, and measured data, the environment in which one uses visualization is a heterogeneous computational environment. Heterogeneity is a fact of life for any organization, where budget, politics, and resources determine the landscape. We felt that for collaborative visualization, it was important to satisfy the judicious movement of computation, networking, and control. The desire to support heterogeneous platforms and varying network capabilities, implies a quality of service goal for the collaboration, with the ideal achievement allowing those with limited resources to participate, while not penalizing those with large resources.

Application goals were to support general development in groupware. The ability to try different forms of floor control, session management, and responsibility partitioning were important as it is not clear where such tasks should belong in our new applications. Additional factors, such as permission control on data sets and access control to sessions also need to be supported. Figure 1 shows a schematic of the goals outlined so far. Tom and Peter are working through the network cloud in a conceptualized shared virtual workspace. The shared workspace uses metaphors of the eye positions of Tom and Peter, on left, and the spray can to highlight data of interest, on right. The visualization primitives in the center are created by the spray can, and the goal is that Tom and Peter can share the can in a fair way, both see the results of the visualization, and have the visualization created from data sets residing at either end.

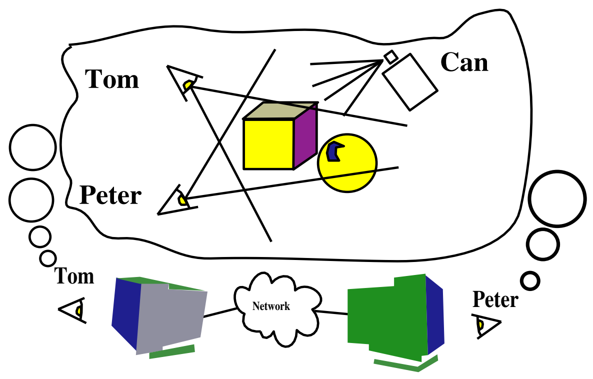


Figure 1: Schematic of a local view from *CSpray* showing Tom's and Peter's eyes, some visualization objects, and a public spray can as they conceptualize the shared workspace while collaborating through the network cloud.

3 Collaboration Aids

The process of collaboratively creating a visualization product in a shared virtual environment has several similarities to a number of group activities such as house construction, playing basketball, or even having a picnic. Each have varying degrees of coordination and sharing, joint or individual efforts, different perspectives, and some common goals. The main logistical difference of these activities with collaborative visualization is the absence of a face-to-face contact in an otherwise interactive group activity. We provide several mechanisms to aid the synchronization, navigation, and collaboration process in a shared virtual workspace.

3.1 Representation of session participants

There are a few basic questions that anyone in a collaborative setting would want to know. Who am I collaborating with? Where are they? And for collaborative visualization, what are they looking at? *CSpray* addresses these concerns with a few straightforward solutions. The active members in a collaborative session are maintained in a pulldown menu – called `Views`. This pulldown menu is updated each time a session participant leaves the session or as new participants join the session. There are two parts to the question of location. Where they are connecting from can be identified from the `Views` pulldown list of session participants. Where they are located within the shared 3D virtual workspace can be observed by their *eyecon*. An eyecon is a 3D icon shaped liked an eyeball showing the location and orientation (see Figure 2) of a session participant within the 3D virtual workspace. Each eyecon also comes with a textual label for the name of the user. Just as the `Views` pulldown is updated dynamically, eyecons are created and broadcasted to announce the arrival of new participants, and are deleted as participants leave the session. The position and orientation of these eyecons are also updated as the participants move around.

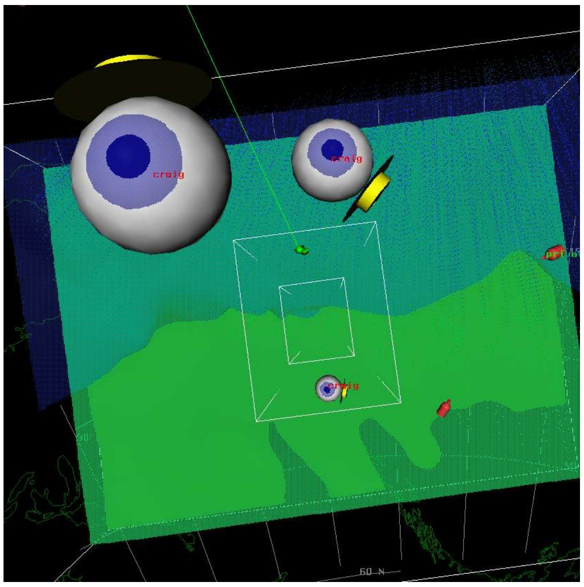


Figure 2: Eyecons to indicate the position, gaze direction and orientation of each session member.

Clicking on another participant's eyecon will transfer his view to your local display. Since eyecons may sometimes not be visible from your current vantage point, view selection may also be done by selecting the user's name from `Views` pulldown. You can then watch what he is doing. If he happens to change his gaze towards you, you will see your own eyecon from his perspective. If he happens to click on your eyecon, he will get the view from your eyecon. Your view will still be from his eyecon.

3.2 Sharing views

CSpray has a popup graphics window (Figure 3). We refer to this new window as the *public* window and refer to the main graphics window as the *private* window. Normally, users see the world from their own perspective through their private window. However, once in a while, they may want to look over their neighbors' shoulder. We provide this What You See Is What I See *WYSIWIS* capability by displaying the your peer's view in your public window.

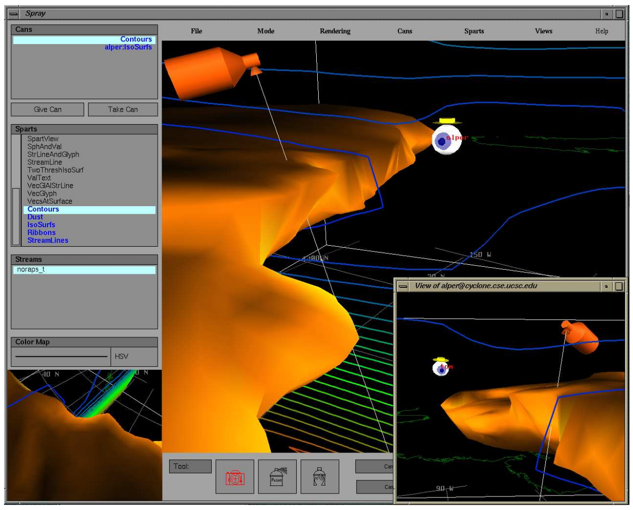


Figure 3: *CSpray* workspace in a two person collaboration. The main window is `Tom's` view. Here, `Alper`'s eyecon is visible. The public window on the lower right contains `Alper`'s view. Here, `Alper` can see `Tom`'s eyecon. The public window allows `Tom` to look over `Alper`'s shoulders.

Another use of the public window is when the group is in a *briefing* session. Here, the attention of the group is on one particular session member. This person is essentially driving the visualization – generating new visualizations, pointing out features, etc. while the group is watching.

Privacy vs consistency

Strictly speaking, *CSpray* does not provide a *WYSIWIS* paradigm. Instead, it is What You See Is What I Let You See *WYSIWILYS*. This difference results from the fact that individual session members have control over parts of the data and visualization they deem are necessary or ready for sharing. That is, the creator of a spray can (and hence the owner of the data set associated with the spray can) has a choice of making the spray can public or private at any time. Only the public cans and their corresponding visualizations are shown in the public windows of other users. Private cans and their corresponding visualizations are kept private and out of other people's views until they are made public. This allows individual users the option to experiment with getting a suitable visualization first as opposed to trying different visualizations while everybody else is watching. This also helps to reduce clutter in the shared virtual workspace.

3.3 3D pointers

Remote cursors are indispensable for collaborative applications with a *WYSIWIS* interface. However, they are inadequate in a shared 3D virtual workspace where each user may have their own views. Our solution is to provide 3D pointers. These are conveniently created by *sparts* that simply draw a 3D arrow. The arrow comes out of the spray can nozzle and is manipulated in exactly the same way as spray cans. We have added two other features into these 3D pointers: One, annotation – the graphical user interface (GUI) that comes with these 3D pointer *sparts* provides a textual interface for users to enter labels to the region of interest. As many 3D labeled arrows can be generated as necessary. Two, synthesized speech – for those who do not have access to an audio input source, we have coupled the annotations to a speech synthesizer. Hence, we sometimes refer to the 3D pointers as annotation cans or talking cans.

3.4 Collaboration state

The main tasks of the *CSpray* session manager are to notify the arrival or departure of session participants, as well as to maintain a consistent collaboration state. When a participant joins a session, the current collaborative state, including the list of current participants, their locations and viewpoints (eyecons), and all the public spray cans and their corresponding visualizations, are transferred to the new participant. In this way, the new arrival is quickly brought up to date on the visualization session. The mechanism to do this is a simple collaboration state save utility. The information stored in the state save reflects the collaboration state at that instant. Any changes to the collaboration state while the state save is being transferred to the new participant are simply queued up as events for the new participant to process.

When a participant leaves a session, a similar set of coordination takes place to maintain a consistent collaborative state among the remaining participants. First, the eyecon and the `Views` menu entry of the person leaving are removed from the windows of the remaining participants. Next, all the public spray cans owned by the person leaving as well as their associated visualizations are also removed from the collaboration. While this may seem rude to the collaborative session, the process is not unlike converting a public spray can back to a private spray can.

3.5 Interactive playback

The same mechanism for saving the collaboration state for accommodating late comers may also be used to save a trace of a collaborative session. This can later be read back and fed to *CSpray* to create a playback of the visualization process. However, unlike a simple playback facility, one can also "collaborate" during playback as if the trace were an active participant in a collaborative session. We find this useful not only as a debugging tool but also for demonstrating the features of *CSpray*. This capability is achieved by the streams interface of *CSpray* for accessing different data sets. The spray cans contain *sparts* which read the data stream(s) on their own. Each data stream is an input to the program. By treating visualization data as output streams, one can record animations. When other event streams are also recorded, one can play back the session.

4 Permissions, Sharing, and Floor Control in CSpray

We address two issues here: overcoming user reluctance in sharing data, and mechanisms for controlling access to shared data.

4.1 Permissions and sharing

There is a tendency for researchers to be protective of their data sets. This may be because of the scarce and expensive nature of their data collection process, or perhaps because the researcher wants to perform some quality checks before releasing it. For whatever reason, we must allay the researchers' concerns regarding who can access, and to what degree someone can access their precious data. The approach of *CSpray* is to provide different levels of data sharing and let the researcher decide what and when the data can be shared. Data sharing can happen at the raw data level, visualization primitive level, or at the image level.

Sharing raw data implies that collaborators trust each other enough to allow the raw data be downloaded before the start of the collaborative session. One limitation of most collaborative visualization systems is that they require this level of trust among users. Depending on the power of individual workstations, this level of data sharing also provides the highest interactive performance in a collaborative session. At the other end, data sharing at the image level allows a very restricted form of collaboration where other people can see the results of the visualization process and request different viewpoints, but not much else. On the other hand, it also allows users with minimal graphics hardware to participate in a collaborative session as observers. In between, there is data sharing through the visualization primitives. Here, the geometric primitives (such as points, lines, polygons) representing the visualizations of their local data sets are shared among the collaborators. The collaborators do not have direct access to the raw data. But may interact indirectly via the spray cans that are connected to the data sets. That is, aside from seeing the visualizations of the data, they may manipulate and (re)spray the cans that generated those visualizations. Since these geometric primitives may get quite large, only the newly created ones are broadcasted to the group. Aside from reducing network traffic, this scheme also simplifies the task of maintaining a consistent collaborative state.

Users have explicit control over how their data are going to be shared. Those attached to spray cans are only shared when the spray can is made public. While private, the spray can and its visualizations are invisible to others. Once public, the spray can and its visualizations are broadcasted to the group. Session members have the opportunity to manipulate and spray any public spray cans to generate new visualizations from remote data sets. Once a user has the floor on a spray can, he also has permission to manipulate parameters associated with the can (nozzle, spray density, etc.) as well as *sparts* parameters (threshold levels, color mapping, etc.). However, only the spray can creator and owner of the data stream has the ability to delete a public spray can.

4.2 Floor control

All public or shared objects and resources may potentially be requested by more than one participant at a given time. Hence, some means of regulating who has access, and the smooth transfer of control of these shared objects, is necessary to avoid contention. This orderly transfer of control of shared objects gives rise to different floor control strategies. In general, each shared object requires separate floor control.

Having the floor on a shared object means that no one else can manipulate the shared object. Hence, the ideal floor control strategy should be fair and ensure that each session member have an equal likelihood of accessing the shared resource. Likewise, they should also be responsive without unduly slowing the transfer of control in an effort to be fair. For example, a "dumb" stop light that controls access to an intersection using fixed time allotments per direction is not perceived as responsive for someone waiting for a green light when there are no cars crossing the intersection. On the other hand, a "smart" stop light that turns green in such situations would be considered more responsive and fair. In fact, this is the floor control strategy in *CSpray*.

In *CSpray*, public spray cans are the only shared objects. Each public can has a floor manager and a transferable floor controller. Floor management is locally *static* because the ownership of the public can always resides on the same machine where it was created. On the other hand, floor control is *distributed* and does not have to be requested from one central source thereby improving the scalability of the number of shared objects. We further distinguish between the floor manager and the floor controller of a public spray can. The manager creates shared objects and holds the data. Control is temporarily granted to any collaborator who requests the shared object. That application then has control, but does not manage the object. The default controller is the manager, or the last person who requested the shared object. Finally, each floor manager is responsible for broadcasting messages to session participants. Figure 4 shows the flow diagram for the floor manager scheme in *CSpray*.

CSpray uses the "smart" stop light analogy for its floor control strategy. It uses a combination of color coded labels (red, yellow, green) to alert users to the status of a shared object and a time out mechanism to free up idle shared objects. Specifically, shared objects under floor control can be in one of four states – *free*, *owned* (by local user), *taken* (by another), and *requested*. Public cans are labeled with the can's current controller, unless the can is *free*, in which case there is no label. To distinguish between the public and private spray cans, private cans are labeled `private` and are always in the *owned* state. The labels of the current controller of the public spray cans are color coded with red (*taken*), yellow (*requested*), and green (*owned*). See Figure 5.

If the can is *taken* or *owned* and then someone requests control of the can (by clicking on the can or selecting it from a menu of available cans), the can becomes *requested*. This alerts the controller that someone wants control without necessarily forcing them to give it up immediately. The current controller may then explicitly release the can, or after some idle time, the can will be implicitly released. Details of how control is passed are illustrated below.

Can movement:

When the floor manager receives a message to move

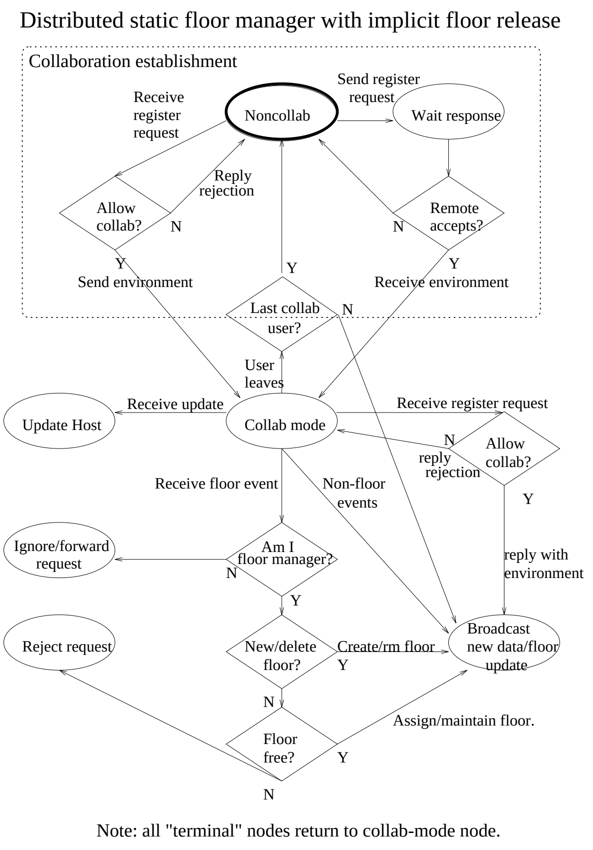


Figure 4: Distributed static floor control management for *CSpray*.

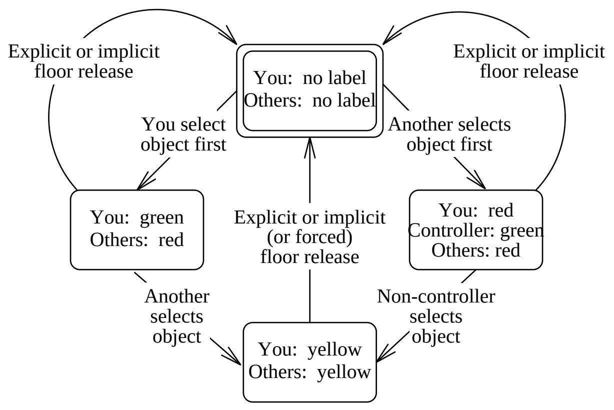
a public spray can, the manager first verifies that it is indeed the floor manager of this can and that the request comes from the can's current controller. If so, the can is moved and the move is broadcast. If the request, comes from someone who does not hold this can's floor, the timer is checked to see when the controller last modified (moved or sprayed) the can. If the timer has expired, control is given to the requester and the can is moved.

Can deletion:

Deletion of a public can is similar to can movement in the sense that a check must first be made to see if somebody else has the floor on the can that is about to be deleted. Aside from timeout requirements, the member must also be the can creator/manager in order to successfully delete the can.

5 Experiences with CSpray

CSpray has been used in a variety of collaborative situations: demonstration, educational, and research. We describe the variety of uses of CSpray, and draw conclusions as to the usefulness and/or extensibility to



Explicit or implicit floor release
You: no label
Others: no label
Explicit or implicit floor release
You select object first
Another selects object first
You: green
Others: red
Explicit or implicit (or forced) floor release
You: red
Controller: green
Others: red
Another selects object
Non-controller selects object
You: yellow
Others: yellow

Figure 5: State diagram illustrating the stop-light analogy floor control strategy in *CSpray*.

support specific future users. *CSpray* has been successfully used to evaluate the concepts and implementation of a collaborative scientific application. By demonstration, and in research settings, we have been able to also show that *CSpray* is a complete application that is useful in communicating visualization results, and investigating data sets with people at remote sites. We discuss the experiences in each type of setting: public demonstrations, educational uses, and research work in visualization and environmental science.

5.1 Public demonstrations

Through a variety of conferences and on campus opportunities we have demonstrated *CSpray* to a large number of people. Our earliest public demonstration was the IEEE Visualization '94 academic demonstration. We interconnected between the Washington, D.C. Visualization'94 conference site, the San Francisco Multimedia'94 conference site, and the University of California, Santa Cruz (UCSC) Laboratory for Visualization and Graphics (SLVG). Demonstrations ran for three days, with traffic being sent over the Internet among the sites, with at most four nodes active at any one time. One difficulty we encountered in setting up a collaborative session was coordinating participants across different time zones. Without using any special network devices beyond getting a basic Internet connection, there was a lag time of about 1-2 seconds between mouse events and activity in the public window.

In 1995, through participation in the CalREN funded, regional ATM (asynchronous transfer mode) testbed network, we had the opportunity to demonstrate *CSpray* at Supercomputing '95 in San Diego. The setup was for four days, running over PacBell's ATM network, connecting an Indigo^2 Extreme at UCSC, to another Indigo^2 Extreme in San Diego. We used a 15 Mbit/sec permanent virtual circuit. We were able to experiment with sending large amounts of visualization primitives through quick updates between the collaborating programs. Having matched nodes, and high bandwidth facilitated a symmetric setup between collaborators. Response was good from the conference attendees. The lag time experienced in this set up was minimal and was comparable to running the collaborative session within a local Ethernet hub.

We also had opportunity in our 1996 Spring Fair – an outreach program for prospective college attendees – to demonstrate *CSpray* on three Indigo XZ's and an SGI Onyx/RE2. Students were invited to sit down and try out the program, or to watch a demonstration of the capabilities. Parents, and young children were given opportunity to try things out as well. Figures 6 and 7 show some of the visualizations generated during the collaborative demonstrations.

The primary lessons we learned from our public demonstrations were the large amount of explanation required, the difficulty in coordinating meetings across the country, the essential fallibility of demo setups, and the different focus of the casual observer. Visualization is not a widely used technology. While visually captivating, the casual observer has a lot of ground to cover before understanding the application and networked applications. We also found it difficult to coordinate setup with our remote collaborators, especially without phone lines during the setup day of the conferences.

5.2 Educational uses

Demonstrations are good for getting visibility of current research, but do not evaluate or use the technology in the fashion it was intended. We also had opportunity to use *CSpray* in educational settings which provides a more accurate assessment of its effectiveness.

In the REINAS project [9], we work with faculty members from the meteorology and oceanography departments at the Naval Postgraduate School (NPS). NPS has a teaching laboratory of Silicon Graphics workstations, in which they used *CSpray* to demonstrate the features of the current day's data. Each student is a participant in the collaboration, and the instructor can effectively drive the visualization. The increasing use of 3D visualization techniques for forecasting makes the demonstration and use of *CSpray* relevant to NPS, and gives students an opportunity to work with and learn about new scientific visualization techniques.

CSpray was developed for REINAS. Important to that project is the development of state of the art tools for environmental sciences. One of the important uses of *CSpray* has been to educate reviewers about the REINAS research challenges and technology. The site review in 1994, was used as an opportunity to test out *CSpray* for educating people about the visualization and meteorological research to date on REINAS. A connection from SLVG to the briefing room at NPS was set up, and the participant at Santa Cruz was able to enlighten the audience as to the power of a realtime connection to a remote participant by using their eyecon to nod and shake their head, a simple yet effective gesture.

In additional to the collaboration and tool elements, *CSpray* has been used as a platform for students wishing to develop extended networking capabilities. For example, in UCSC's networking courses, *CSpray* has

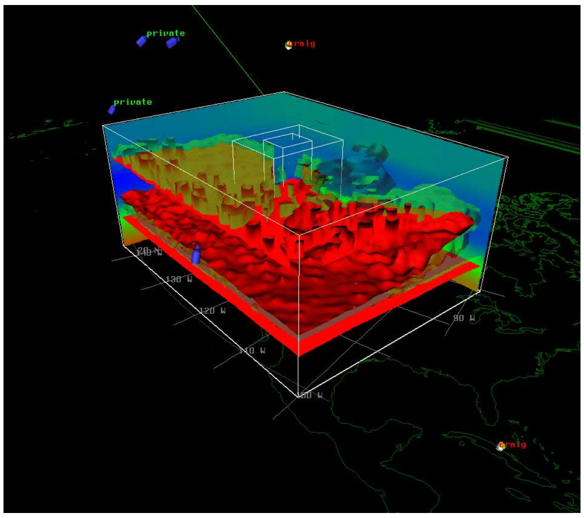


Figure 6: Complex isosurface visualizations of humidity and temperature fields.

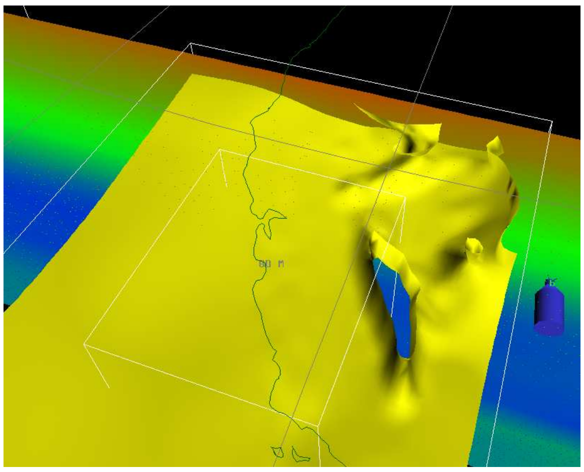


Figure 7: Investigation of a hole in the humidity isosurface field over the hot and dry central California valley.

been used as a foundation for students to develop libraries for reliable multicast, and try them out with a real application. This exposes students to the issues in supporting applications, teaches them about collaborative applications, and also provides a driving force for creating efficient networking libraries and techniques.

The primary lessons learned from our educational uses of *CSpray* have been the acceptance, complexity of the user interface, reliability problems, and difficulty in code reuse. Of course the acceptance is helped by our hands on tutorials and assistance, and difficulties with the user interface can be somewhat mitigated through help as well. Developing a complex application that is reliable and easy to use is difficult, and a large amount of effort has been expended to meet these goals.

5.3 Research uses

CSpray was developed as a research tool and has made a direct impact in research in the fields of meteorology, biology, and oceanography. In meteorology, *CSpray* has been crafted for visualization of 3D atmospheric simulation models. In biology, new data collection activities of data from instrumented elephant seals, has been supported by importing the data sets, and developing new analysis tools for them. Regional research in oceanography is supported through visualization of real-time collected ocean data. Work on how to visualize uncertainty in such ocean data has also resulted from this collaboration [10].

Other related research projects have spun off from *CSpray*, including compression research. Because of the slower bandwidth links used to connect through the Internet, savings achieved through compression can be substantial. We have investigated the compression of frames of images to be communicated, and traded this off to the compression of the visualization primitives themselves. Currently an active area of research [11], the compression and transmission of geometric primitives is important for fast inexpensive collaborative scientific visualization. For a summary of compression efforts related to *CSpray*, see [12].

Currently, the collaboration layer in *CSpray* is tightly coupled with the rest of the visualization system. While there are other efforts in providing collaboration-unaware systems, they also create difficulties with regards to floor control issues, and effective collaboration aids. An alternative might be a collaboration-ready system where a well-defined, application independent, collaboration layer supports the needs of coordinating events and data streams among session participants. Until network speeds are high enough and network costs are not a primary concern, there is also a need to look at a couple of related issues:

supporting clients without hardware rendering and/or very low network bandwidth. More work is needed on matching service levels according to client's resources.

6 Conclusions

We presented the architecture and components of *CSpray* including a host of collaboration aids, recording and playback facility of collaborative sessions, built in session management, and an intuitive and fair floor control scheme. *CSpray* provides a group of geographically distributed scientists the means of sharing their data, while maintaining exclusivity, and interactively creating and analyzing visualization products.

We also discussed the variety of experiences in educational, scientific, and research settings for *CSpray*. We described the effective interaction available for exchange and communication of scientific visualization products as they are created. The basic problems of collaboration are yet to be solved, how to create work groups, communicate, provide efficient scalable services, etc. What we have presented is a collection of effective and intuitive design decisions and an architecture that have proven to be effective in a wide variety of circumstances.

For our research goals, *CSpray* has been tremendously successful as it has given us the ability to evaluate different collaborative approaches. The concerns of where to place the services, such as session management and control are somewhat separate from simply creating, developing, and evaluating collaborative software for particular applications. The services location question becomes much more important for synthesized collaborations of video, voice, visualizations, and whiteboards. We hope that our developments in collaborative scientific visualization provide a baseline from which to compare for the more highly integrated environments.

Our research directions include the development of more aggressive combined multimedia streams, methods to communicate them, and means for heterogeneous processing and resource allocation to get the highest performance possible out of particular network(s) and workstations. By experimenting with ATM, Ethernet, and wireless networks, we have been able evaluate the impact of a large disparity of available bandwidths. Also, by having access to many users in a cross disciplinary regional collaborations we have been able to assess the effectiveness of metropolitan area collaborations, and work with a large range of workstation capabilities. We shall focus on the creation of better tools, performing more conclusive evaluations of tools, and incorporation of faster networks, faster computers, and additional media types all to make collaborative scientific computing an effective agent for better group collaboration.

Acknowledgements

We would like to thank students and former students who helped develop the *CSpray* system including Naim Alper, Jeff Furman, Tom Goodman, and Elijah Saxon. We would also like to thank our user base who provided early inputs and evaluation feedbacks to drive the development of *CSpray* including Dan Fernandez, J.J. Garcia-Luna, Bruce Gritton, Paul Hirschberg, Glenn Langdon, Wendell Nuss, Guy Oliver, and Jeff Paduan. The Santa Cruz Laboratory for Visualization and Graphics (SLVG) provided a wonderful research environment and we would like to acknowledge that. Finally, the authors would also like to acknowledge support from ONR grant N00014-92-J-1807, NSF grant IRI-9423881, NASA Cooperative Agreement NCC2-5176, and PacBell for the CalREN BayNet ATM funding.

References

[1] David Marca and Geoffrey Bock. *GROUPWARE: Software for computer-supported cooperative work.* IEEE Computer Society Press, 1992.

[2] M.R. Macedonia, et al. Exploiting reality with multicast groups: a network architecture for large-scale virtual environments. In *Proceedings of Virtual Reality Annual International Symposium '95*, pages 2–10, March 1995. Los Alamitos, CA, USA: IEEE Comput. Soc. Press.

[3] J.L. Encarnacao, B. Tritsch, and C. Hornung. DEDICATED - learning on networked multimedia platforms. In *Visualization in Scientific Computing: Uses in University Education. IFIP WG3.2 Working Conference*, volume A-48, pages 67–75, July 1993. Irvine, CA, USA, 28-30 July 1993.

[4] V. Anupam and C. Bajaj. Shastra: multimedia collaborative design environment. *IEEE Multimedia*, 1(2):39–49, Summer 1994.

[5] H.-G. Pagendarm. Unsteady phenomena, hypersonic flows and co-operative flow visualization in aerospace research. In *Proceedings Visualization '93*, pages 370–373, 1993. San Jose, CA, USA, 25-29 Oct. 1993. Los Alamitos, CA, USA: IEEE Comput. Soc. Press.

[6] M. J. Gerald-Yamasaki. Cooperative visualization of computational fluid dynamics. *Proceedings of the Eurographics'93*, 12(3):497–508, September 1993.

[7] Jean Clucas and Velvin Watson. Interactive visualization of computational fluid dynamics using Mosaic. In *Second International WWW Conference*, October 1994.

[8] A.S. Jacobson, A.L. Berkin, and M.N. Orton. LinkWinds: interactive scientific data analysis and visualization. *Communications of the ACM*, 37(4):42–52, April 1994.

[9] D.D.E. Long, et al. REINAS the real-time environmental information network and analysis system. In *Proceedings of COMPCON*, pages 482–487, San Francisco, CA, March 1995. IEEE.

[10] Craig M. Wittenbrink, Alex T. Pang, and Suresh K. Lodha. Glyphs for visualizing uncertainty in vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):266–279, September 1996.

[11] Michael Deering. Geometry compression. In *Proceedings of SIGGRAPH 95*, pages 13–20, August 1995.

[12] Glen G. Langdon, et al. Compression research on the REINAS project. In *Workshop on Science Information Management and Data Compression*, NASA Goddard Space Flight Center, Greenbelt, MD, October 1995.

Alex Pang

Alex Pang is an Assistant Professor in the Computer Sciences Department at University of California, Santa Cruz. He obtained his MS and PhD in Computer Science from UCLA in 1984 and 1990. His current research interests are in collaboration software, uncertainty visualization, scientific visualization, and virtual reality interfaces. More information from http://www.cse.ucsc.edu/ pang

Craig Wittenbrink

Craig M. Wittenbrink is a research engineer at Hewlett Packard Laboratories. After working as a post doctoral researcher and lecturer at University of California, Santa Cruz doing research in environmntal visualization from 1994-1996, he accepted a position at Hewlett Packard in order to get more opportunity to implement his computer graphics hardware architectures and ideas. He received a B.S. in 1987 in electrcal engineering and computer science from the University of Colorado, his M.S. in 1990 and Ph.D. in 1993 in electrical engineering from the University of Washington. He was a design engineer with Boeing Aerospace from 1987 to 1989 where he developed computer image generators. He was a co-chair of the 1995 IEEE/ACM Parallel Rendering Symposium, and is widely published in parallel–image, visualization, and graphics processing. He is a member of ACM, Siggraph, and the IEEE Computer Society.