CSpray:
A Collaborative Scientific Visualization Application

Alex Pang, Craig M. Wittenbrink and Tom Goodman

Baskin Center for Computer Engineering & Information Sciences
University of California, Santa Cruz

Santa Cruz, CA 95064

ABSTRACT

We present the design and implementation of Collaborative Spray or *CSpray* (pronounced "sea spray"). *CSpray* is a CSCW (Computer Supported Cooperative Work) application geared towards supporting multiple users in a collaborative scientific visualization setting. Scientists are allowed to share data sets, graphics primitives, images, and create visualization products within a view independent shared workspace. *CSpray* supports incremental updates to reduce network traffic, separates large data streams from smaller command streams with a two level communication strategy, provides different service levels according to client's resources, enforces permissions for different levels of sharing, distinguishes private from public resources, and provides multiple fair and intuitive floor control schemes for shared objects. Off the shelf multimedia tools such as nv and vat can be used concurrently. *CSpray* is based on the *spray rendering* visualization interaction technique to generate contours, surfaces, particles, and other graphics primitives from scientific data sets such as those found in oceanography and meteorology.

Keywords: computer supported cooperative work, collaboration, scientific visualization, spray rendering, floor control, session management.

1 INTRODUCTION

Recent developments in teleconferencing allow geographically separated colleagues to discuss ideas using voice and video in real time. There are many systems designed to support multimedia interaction among users[1–7] or study the mechanisms behind multimedia interaction. However, with a few exceptions, most visualization systems to date still operate in single-user mode. With single-user tools, visualizations 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 *collaborative scientific visualization* settings allow multiple investigators to share data, views, manipulation sequences and to participate in the creation of the visualization products across the network.

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 our new solutions for a scientific collaboration environment that provides:

- Synchronized workspaces for group generated visualizations.
- Private workspaces for visualizations generated independently of a collaboration group.
- Local data integrity, protection, and independence from the collaboration.
- Resource allocation, such as decompression, graphics rendering, and routing.
- Reliable and unimposing floor control schemes to manage access to shared resources.
- A user-interface (UI) that reduces problems of screen clutter, and clearly shows: participants (who is where), activity, and ownership (who is controlling what).

2 COLLABORATIVE SPRAY RENDERING (*CSpray*)

CSpray stands for Collaborative Spray rendering and is an extension of *Spray*, a visualization application, into a collaborative environment. We first give an overview of *Spray*. We then discuss the changes necessary to support a collaborative visualization environment.

2.1 Spray rendering

Spray rendering[8,9] is a framework that we and our colleagues have developed for visualization which uses a spray can metaphor; cans are filled with smart paint particles (*sparts*) that are sprayed into the data to highlight interesting features. Features are displayed when sparts become activated and leave visualization objects in their path. These visual objects may be lines, polygons, spheres, and other graphics primitives that delineate the data set under study. Collectively, these are called abstract visualization objects (*AVO*s).

Users have control over several parameters. To aid in the delivery of sparts into the data space, users can adjust beam focus, can position, and direction. The number of sparts delivered per dose, their spatial distribution, and whether the can is in probe mode (AVO present only where can is pointing) or normal spray mode (AVOs stick around) can also be specified. Other parameters such as sparts' paths, lifetimes, colors, and types of AVOs can all be customized as part of the spart's definition.

Figure 1 shows the effects of spraying several cans in different directions to extract an isosurface, a dust cloud, and stream vectors from a test data set. The currently selected spray can is in the upper right of the window, and looks like a shaving cream can with a rod coming out of it. At the end of the rod is a sphere which indicates the center of view of the can. The view from that can is shown in the smaller graphics window in the lower left.

Spray deals with a streams oriented application programming interface (API) to retrieve data from remote database servers. The API can also handle event streams from user interactions and output streams of AVO visualization products. The stream interface is quite flexible and can be used to record animations as AVOs are generated or can be used to play back a session by recording the event streams. With slight modifications, we also use the same API for transfer of events, requests, data, AVOs, and images among collaborators in a *CSpray* session.

2.2 Components of *CSpray*

Several modifications were added to *Spray* in order for it to support collaboration. In *CSpray*, several users in a visualization session may analyze a set of distributed data by creating spray cans loaded with sparts. Sparts are tied to one or more local or remote data sets. Spray cans may be made public or kept private. Public cans are visible and accessible by other participants. Participants can see the spraying action of other users in their local window. More than one participant may be spraying at any given time. Once in a while, the attention of the entire group may be directed at what one of the participants is doing. Participants may also join and leave the session at any time. Figure 2 shows the schematic local view of a *CSpray* participant and the eyes representing two other collaborators, a spray can, and a couple of visualization objects.

Starting *CSpray*

CSpray may be started in standalone or collaborative mode. In standalone mode, *CSpray* behaves like the single-user application *Spray*. In collaborative mode, *CSpray* passes relevant information to and from other participants and maintains a dynamic list of active participants.

Spray cans and Permissions

Spray cans contain sparts that are associated with one or more data streams. Therefore, other participants may be allowed indirect access to local data through the AVOs generated by using those spray cans. To accommodate participants who wish to limit access to some of their local data, we differentiate *private* and *public* spray cans. A can is private by default. Once the creator decides to make it public, the can and its AVOs becomes visible to all other participants. Participants can then take turn to grab, manipulate, and spray the can. Deletion of public cans is allowed only by the creator and owner(s) of the data stream(s) tied to the spray can.

Thus, each user's shelf of spray cans consists of their own private can collection and a common public can collection. The use of a public spray can is translated into requests to the host machine that owns the can to generate and send the appropriate AVOs. It is also possible for a spray can to have multiple data streams from different hosts. In that case, can use implies requests to several remote hosts for access to different data streams.

CSpray has one additional public can type that is used as a 3D remote cursor. This *pointer can* indicates interesting features to another person in the shared workspace. Pointer cans resemble spray cans except they have no input stream.

Floor Control

All objects that are public may potentially be grabbed by more than one participant. Therefore, floor control or some means of regulating who has control on a public object, is necessary to avoid contention. In *CSpray* public spray cans and pointer cans are examples of public objects. Whoever has the floor on a public can may grab and spray (or point) it. The floor control for a can is handled by timed release where the can is free to be grabbed after a certain amount of idle time. Other alternatives investigated included explicit release, where the current holder needs to acknowledge/deny requests for a can; and explicit control where the holder has the can only while holding down some combination of keys or mouse buttons.

Objects under floor control will always 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. (In *CSpray* private cans are labeled private and they are always in the *owned* state). We color code the labels and the cans with red, yellow, and green (analogous to a stop–light) to represent the states: taken, requested, and mine.

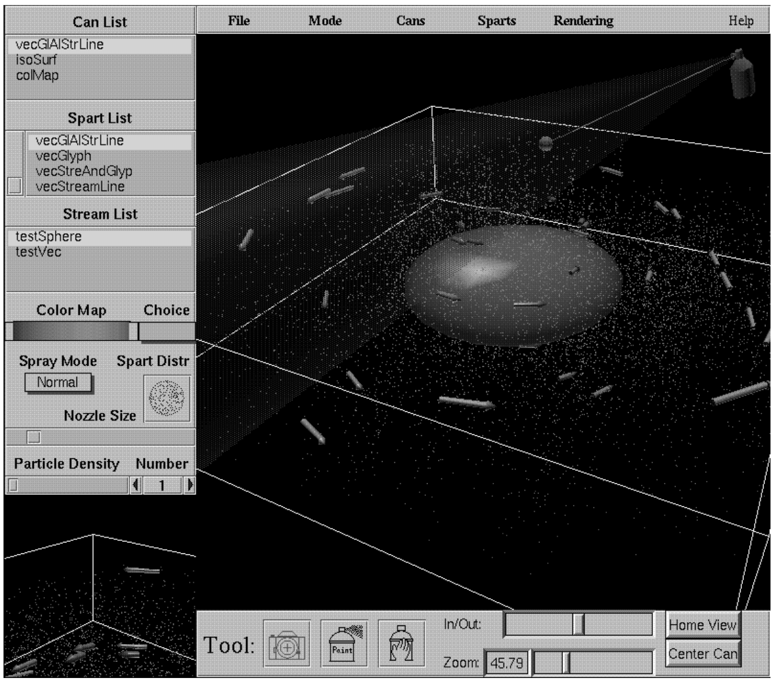

Figure 1: Spray rendering workspace showing effects of different types of smart particles (*sparts*). Users control viewing, can position, can orientation, and spraying through either graphics window. The lower left graphics window shows the view from the current can.

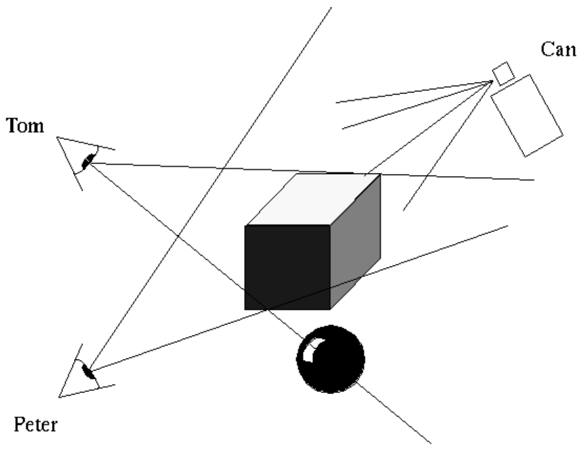

Figure 2: Schematic of a local view from *CSpray* showing Tom's and Peter's eyes, some visualization objects, and a public spray can.

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. Another option that is also available is to force floor release if someone is keeping a requested object for too long.

Eyecons

Eyecons represent the presence and position of other participants in a session. They are literally modeled as floating eyeballs as shown schematically in Figure 2 and operationally in Figure 4. The eyecon is labeled with the login name of the participant for easy identification.

Public View Window

CSpray has an additional popup graphics window (Figure 4). We refer to this new window as the *public* window and to the main graphics window (Figure 1) as the *private* window. Normally, users see the world from their own perspective. However, once in a while, they may want to look over their neighbor's shoulder and *snoop* on what the neighbor is doing. By clicking on an eyecon or by explicitly selecting a participant's name from a dynamic `Views` pulldown menu, a user specifies the person whose view will appear in the public window. The view from that eyecon (excluding all private objects) is then displayed in this public window. Information to generate the remote views are transferred through streams. Thus, changing views is simply a matter of changing stream connections.

The public window is commonly used in *briefing* mode. This is where the group's attention is focused on a single presenter's view and actions. Observers will all select the presenter's view. The presenter may wish to select his own name from the views menu, allowing him to observe what others are viewing – his own public view. This view is useful to the presenter, since it visually confirms which objects are public (visible) and which are private (invisible). You can let other participants know that you want to show them something in the public window via available audio tools such as vat[10] .

Limiting Network Traffic and Visual Clutter

Permission allows or prevents other members from spraying into your window. This controls whether you want to receive the AVOs generated by a particular can. By default, private cans do not send their can orientation, movements, or AVOs to other participants' private or public windows, while public cans send this information to both windows. Currently, *CSpray* allows the local user to toggle on and off the viewing of any can (private or public) and its AVOs. This provides a quick way to reduce clutter in the workspace. In a networked environment with different levels of service, a participant may be a particularly slow node. Users may toggle the AVOs from other users on or off to improve interaction time. Built in flow control and varying rendering update rates are also used to match participants capacities.

Sharing

CSpray allows participants to create visualization products through data sharing. This happens at the data level, AVO level, or image level. As permissions allow, optimizations may take place transparent to the application. Data streams access is a remote file or database access. Therefore, stream connections are given only to clients with sufficient network bandwidth. Data replication at different sites provides higher interactive performance after an initial data transfer. However, if the data is private or is constantly changing, the raw data is not distributed, and only the AVOs are shared instead. Since the number of AVOs can be quite large, only the newly created AVOs are broadcasted. In addition, different levels of rendering and network capabilities require applications to adjust their interactive performance by switching from sending AVOs to sending rendered images. Users still grab and use cans, but they are simply getting rendered images as opposed to a list of AVOs or replicated data streams.

Joining and leaving a session

Participants can join a session at any time. When joins occur, the current state of AVOs, users and viewpoints, public spray cans, and stream attachments are transferred to the new participant. Information about the new participant is also announced to the other session members. *CSpray* supports these functions for bringing late comers up to date on what has occured. Participants may join a session as a fully active and contributing member or as a passive observer.

Just as users may join at any time, they may also leave at any time. Leaving a session requires coordination with the *CSpray* clients to disengage data sets, cans that were controlled by the users, and giving up floor control for other objects.

3 DESIGN AND IMPLEMENTATION OF *CSpray*

Session management and floor control functions are implemented in the application itself, making it *collaboration-aware.*

3.1 Architecture of *CSpray*

CSpray uses a symmetrical client server collaboration model. Each collaborative host runs a local copy of *CSpray.* BSD (Berkeley Software Distribution) UNIX sockets are used for network communication. Two sockets are opened to every collaborator. A reliable TCP/IP (transmission control protocol/internet protocol) socket is used for control information, and another socket is used for transmission of data and positional update events.

Among comparably equipped collaborators, *CSpray* transmits geometric data (AVO) to all participants. This method makes use of each machine's power to process data and display images, unlike in a strict client server model. By only sending the AVOs, collaborators are able to keep their individual data private. This allows scientists to share the results of their visualizations without granting full access to the raw data. Since the data need only exist on one collaborating host, there is no further concern for tracking updates of multiple copies of data files.

Unlike in the framebuffer data transmission method, another benefit of this model is that it allows each participant to view the data independently, from any perspective, without requesting additional information from the host where the data is stored. This is possible because the view-independent AVOs resides on every host. By transmitting a participant's view matrix, views can be shared with all participants. With the framebuffer method, this would require compressing and transmitting the entire framebuffer, instead of the 4x4 floating point value view matrix. Of course, frame only clients are used for workstations or X-stations with insufficient rendering speed.

3.2 Session management

CSpray provides very limited session management, supports late comers, and maintains the session list explicitly. This is done with a linked list data structure called `collab-list` which maintains a list of users in the session. Each record in the collab-list contains a collaborator's name, process identifier, host name, viewing location, and viewing orientation. These information allow *CSpray* to uniquely identify each participant in the session. This identification is contained in every packet transmitted and is necessary when events are received to determine who has sent the event. The viewing information is used to determine eyecon locations and when one participant wishes to track another participant's view.

3.3 Floor control

In *CSpray*, the floor management is locally *static* because the ownership of, for example, a spray can always resides on the same machine where it was created. Alternatively, other participants can grab the floor of the spray can and move or manipulate it, hence it is also *distributed*. Each floor manager knows how to broadcast messages to all participants that have permission to receive data.

We distinguish between the floor manager and the floor controller. The manager creates shared objects and holds the data. Control is temporarily granted to any connected collaborator who requests to work on the public data. The default controller is the manager. Every shared object, at any point in time, has a manager and a controller and if nobody requests the floor the last controller maintains the floor until it is requested or a timer has elapsed.

Rather than having one host which manages all the cans, each version of *CSpray* can manage a set of cans. This distributed floor management scheme avoids consulting a central host for every floor control transaction. We believe distributed control provides greater scalability and fault tolerance since it distributes network traffic and floor control responsibilities. Figure 3 shows a flow diagram for the floor manager scheme for *CSpray*.

3.4 Streams and events

Events in *CSpray* are handled by a message passing paradigm whereby messages are distributed to the various modules within a local host as well as sent to and received from collaborating remote versions. We treat a sequence of events as streams with handles that *CSpray* controls and redirects according to the requirement and classification of the event. Events may be local and therefore non-collaborative or they may be collaborative. The latter type may or may not require floor control. Collaborative events requiring floor control include use of public resources like public spray cans and pointers. When one changes eye point location, this generates a collaborative event which does not require floor control (since no one can force anyone else to move). It is collaborative, because the new location must be broadcast to other participants so eyecon locations can be updated. *CSpray*'s input and output streams are redirected to choreograph the collaboration.

4 COMPARISON WITH RELATED WORK

The main difference between prior work (e.g. CECED[11], SHASTRA[12], HIGHEND[13], BERKOM[14], Tempus fugit[16], Klinker[15], and Collage[17]) and our design approach is that our system supports scientific collaboration

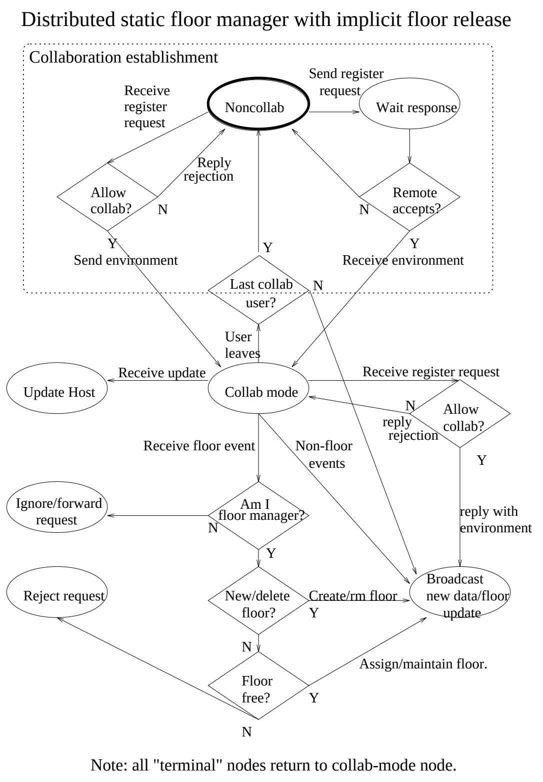

Figure 3: Distributed static floor control management for *CSpray*.

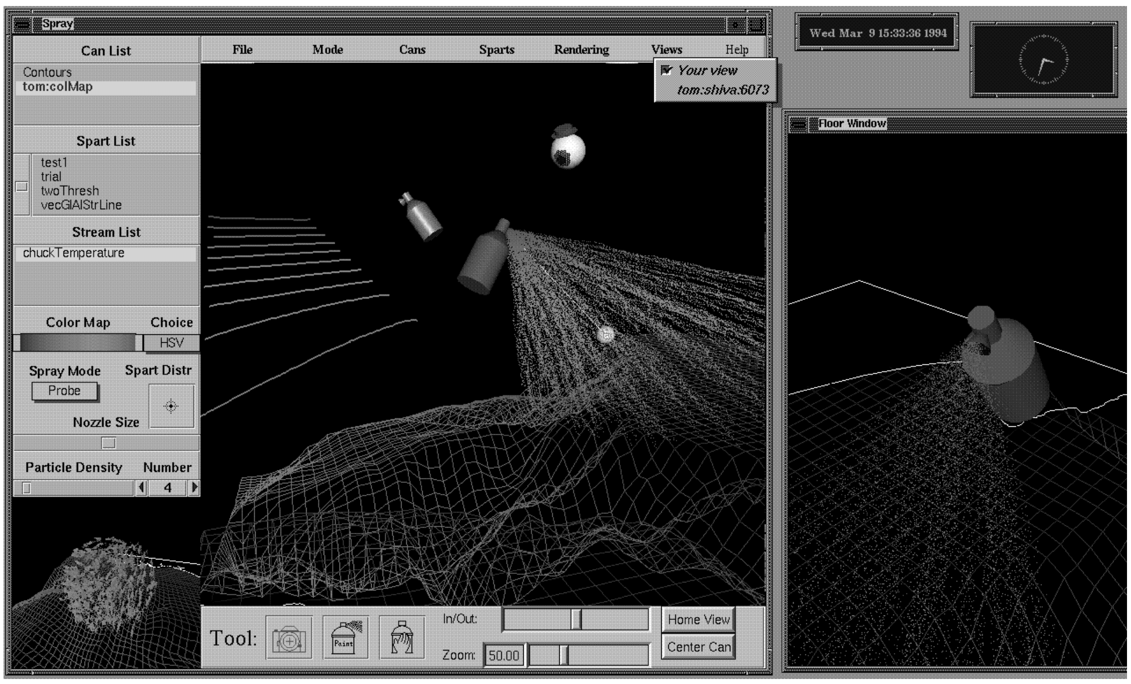

Figure 4: *CSpray* workspace in a two person collaboration. The left window shows the local view, while the right window shows the remote participant's view. Note the remote participant's eyecon in the local view.

with distributed data sets. In contrast, prior work has focused on either using single-user collaboration-unaware applications or developing collaboration-aware applications and tools that exchange simpler streams such as video, audio, white board or pixel maps. *CSpray* provides support for collaborations with AVOs and also supports new collaborative interactive objects such as eyecons, shareable spray cans, name labels, intuitive floor control, and the protocol for exchanging packets describing all of these objects and their movement.

5 SUMMARY AND CONCLUSIONS

We presented the design issues and identified the components of an integrated collaborative scientific visualization software application. Participants in *CSpray* can collaborate and interactively create visualization products. *CSpray*'s new visualization collaboration features are: allowing scientists to indirectly share data and still maintain exclusivity, recording session activities for playback and review, and providing clear handles through eyecons and spray cans for being aware of and interacting with remote users independent of audio and video tools. *CSpray* is a collaboration-aware implementation where permissions are enforced on different levels of sharing.

Additional enhancements currently under development include: matching service levels according to client's resources; compression of visualization streams; porting AVO handling routines to OpenGL for non-SGI platforms; and supporting clients without hardware rendering. Details and updates can be found in our world wide web site under the REINAS research project[18] .

ACKNOWLEDGEMENTS

We would like to thank those who helped develop *Spray* including Naim Alper, Jeff Furman, and Elijah Saxon. Inputs from J.J. Garcia-Luna and Peter Dommel on session management and floor control are also gratefully acknowledged. We would also like to thank those scientists whose input and needs have helped to drive the development of *Spray*and *CSpray*, including Bruce Gritton, Professor Wendell Nuss, Dr. Paul Hirschberg, and Professor Jeff Paduan. This work is funded in part by ONR grant N00014-92-J-1807.

6 REFERENCES

[1] S. A. Bly, S. R. Harrison, and S. Irwin. Media Spaces: Bringing people together in a video, audio and computing environment. *CACM – Special Issue on Multimedia in the Workplace*, 36 No. 1:28–44, January 1993.

[2] M. A. Stefik et al. Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. In *Computer Supported Cooperative Work: A Book of Readings*, pages 335–366. Morgan-Kaufman, 1988.

[3] K. A. Lantz. An experiment in integrated multimedia conferencing. In *Computer Supported Cooperative Work: A Book of Readings*, pages 533–556. Morgan-Kaufman, 1988.

[4] S. Sarin and I. Greif. Computer based real-time conferencing systems. In *Computer Supported Cooperative Work: A Book of Readings*, pages 397–420. Morgan-Kaufman, 1988.

[5] W. H. Mansfield Jr. K, C. Lee and A. P. Sheth. A framework of controlling cooperative agents. *IEEE Computer*, pages 8–16, July 1993.

[6] S. B. Wilbur. Dimensions of sharing in multimedia desktop conferencing. In *IEEE Colloquium on CSCW: Computer-Supported Cooperative Work*, pages 4/1–4, 1990.

[7] H. Smith, S. Benford, H. Howidy, and A. Shepherd. The GRACE project: towards large scale group communication systems. In *IEEE Colloquium on CSCW: Computer-Supported Cooperative Work*, pages 5/1–4, 1990.

[8] Alex Pang and Kyle Smith. Spray rendering: Visualization using smart particles. In *Proceedings: Visualization '93*, pages 283 – 290. IEEE Computer Society, 1993.

[9] Alex Pang and Naim Alper. Mix&Match: A construction kit for visualization. In *Proceedings: Visualization '94*, pages 302 – 309. IEEE Computer Society, 1994.

[10] M. Macedonia and D. Brutzman. MBone provides audio and video across the internet. *CACM*, 27, No.4:30–36, April 1994.

[11] E. Craighill, R. Lang, M. Fong, and K. Skinner. CECED: A system for informal multimedia collaboration. In *Proc. of the ACM 1993 Multimedia Conference*, Anaheim, CA, August 1993.

[12] V. Anupam and C. Bajaj. Collaborative multimedia scientific design in SHASTRA. In *Proc. of the ACM Conference on Multimedia Systems*, pages 447–480, August 1993.

[13] H.-G. Pagendarm and B. Walter. A prototype of a cooperative visualization workplace for the aerodynamicist. In *Proc. of the Eurographics'93*, volume 12, No. 3, pages 485–508, 1993.

[14] M. Altenhofen et al. The BERKOM multimedia collaboration service. In *Proc. of the ACM 1993 Multimedia Conference*, pages 457–462, August 1993.

[15] G. Klinker. An environment for telecollaborative data exploration. In *Proc. of the IEEE Conference on Visualization*, pages 110–117, 1993.

[16] M. J. Gerald-Yamasaki. Cooperative visualization of computational fluid dynamics. In *Proc. of the Eurographics'93*, volume 12, No. 3, pages 497–508, 1993.

[17] National Center for Supercomputing Applications. NCSA-collage. World Wide Web, URL: http://www.ncsa.uiuc.edu/SDG/Software/XCollage/collage.html, 1994.

[18] REINAS. Real time Environmental Information Network and Analysis System. World Wide Web, URL: http://www.cse.ucsc.edu/research, 1994.