[Warshauer] CU SEE-Me: Happenings, Theater, Magic. Susan Warshauer, University of Texas at Austin.

[VRML FAQ] A Window on Shared Virtual Environments. Denis Amselem, SRI International. VRML Frequently Asked Questions (FAQ)

[GNN FAQ] Globewide Network Academy Frenquently Asked Questions (FAQ)

[Gomez, Pea 1992] Learning through collaborative visualization: Shared technology learning environments for science. Pea, R.D. Northwestern University & Gomez, L. Bellcore. 1992.

[Gustafson] Transitioning to Collaborative Groupware Environments. Paul Gustafson. CSC Vanguard, Computer Sciences Corporation.

[Henderson] Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-based GUI. D.A Henderson, S.K. Card. Xerox Palo Alto Research Center (PARC).

[Kindberg] Mushroom: A framework for collaboration and interaction across the Internet. Tim Kindberg. Queen Mary & Westfield College, University of London

[Lejter 1997] Collaborative Mocha user's guide. Luis Lejter, Brown University.

[MBONE FAQ] MBONE Frequently Asked Questions (FAQ)

[Meyer, Blar, Hader] A MOO-based Collaborative Hypermedia System for WWW. Tom Meyer, Brown University. David Blair, Suzanne Hader, Brown University.

[MUD FAQ] MUD Frequently Asked Questions (FAQ)

[Parnes et al] WebDesk: The Tele-Conferencing Service of MATES. Peter Parnes, Dick Schefstrom, Kare Synnes. University of Lulea

[Reinhardt] New Ways to Learn. Andy Reinhardt, Byte magazine.

[Riel 1990] Cooperative Learning Through Telecommunications. Margaret Riel. 1990. The AT&T Learning Network

[SICS] Human-Computer Interaction for Distributed Systems. Swedish Institute of Computer Science (SICS)

[Stefik et al] Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings. M. Stefik, G. Foster, G. Bobrow, K. Kahn, S. Lanning, L. Suchman. Xerox Palo Alto Research Center (PARC).

[Stefik et al] WYSIWIS Revised: Early Experiences with Multiuser Interfaces. M. Stefik, G. Bobrow,G. Foster, S. Lanning, D.Tatar. Xerox Palo Alto Research Center (PARC).

[Tamassia, et al.] Algorithm Animation over the World Wide Web. James E.Baker, Isabel F.Cruz, Giuseppe Liotta, Roberto Tamassia. Brown University

[Woo, Rees] A Synchronous Collaboration Tool for World Wide Web. Tak K. Woo, Michael J. Rees. The University of Queensland

9.0 Bibliography

[Bajaj 1995] Collaborative Multimedia & Scientific Toolkits. Chandrajit L. Bajaj, Department of Computer Science. Purdue University

[Benford et al] Managing Mutual Awareness in Collaborative Virtual Environments. Steve Benford, The University of Nottingham. John Bowers, The University of Manchester. Lennarte E. Fahlen, The Swedish Institute of Computer Science (SICS). Chris Greenhalgh, The University of Nottingham

[Benford, Fahlen] Viewpoints, Actionpoints and Spatial frames for Collaborative User Interfaces. Steve Benford, The University of Nottingham. Lennart E. Fahlen, Swedish Institute of Computer Science SICS

[Brikcer et al] Multiplayer Activities that Develop Mathematical Coordination. Lauren J. Brikcer, Steven L.Tanimoto, Alex I.Rothenberg, Danny C. Hutama, Tina H. Wong. University of Washington

[Conner et al] Making Telecollaboration as Real as Being There. Brook Conner, Greg Turk, Jonathan Corson-Rikert. NFS-STC Computer Graphics and Scientific Visualization Center

[Curtis] LamdaMOO Programmer's Guide. Pavel Curtis. Xerox Palo Alto Research Center.

[Curtis et Al.] The Jupiter Architecture: Secure Multimedia in Network Places. P. Curtis, M. Dixon, R. Frederick, D.A. Nichols. Xerox Palo Alto Research Center

[Curtis et al] The Astro VR Collaboratory, An On-line Multi-User Environment for Research in Astrophysics. D. Van Buren, Infrared Processing and Analysis Center. P. Curtis, D.A. Nichols, Xerox Palo Alto Research Center. M. Brundage, University of Washington

[CU-SEEME FAQ] CU SEE-Me Frequently Asked Questions (FAQ)

[Frivold et al] Extending the WWW for Synchronous Collaboration. Thane J.Frivold, Ruth E. Lang, Martin W. Fong. SRI International

[Gajewska et al.] Argo: a System for Distributed Collaboration. Hania Gajewska, Jay Kistler, Mark S. Manasse, David D. Redell. System Research Center, Digital Equipment Corporation

[Gamma et al] Design Patterns. Erich Gamma, Richard Helm, Ralph Johnson, John Vissides. Addison Wesley.

[Gleeson et al] Beyond HyperText: Using the World Wide Web for Interactive Applications. Martin Gleeson, Tina Westaway. The University of Melbourne

oped by Microsoft based on OLE. Java Beans will have the advantage of Java's platform independence, and the security provided by the Java VM. Active X is powerful, but it is currently limited to the Intel/windows 95/NT platform and since it sends binary code over the net it presents serious security questions.

By redesigning Mocha in terms of components we can simplify the development of Mocha applications and add collaboration and algorithm animation features to existing applications as well.

8.6 Multi-threaded Server

The Collaborative Server implemented in CMocha is single-threaded. It uses the "poll" UNIX mechanism to detect data waiting to be read and avoid blocking its socket connections. This design decision simplified the overall implementation of the Server. However it presents the drawback that a lengthy operation in a particular algorithm animation execution can produce a noticeable "lag" in the feedback to other clients running on the server at the same time This problem can be solved by implementing a fully multi-threaded Server.

8.7 Extend course materials and work based on Mocha

As section 7 described, Mocha can be a used as an educational aid in a variety of situations. These situations include a co-present environment, namely CS252 here in Brown's CS Department, and also learning-at-distance and cooperative-learning-at-distance schemes.

The most attractive of these potential uses of Mocha is its use as an educational aid in CS252, the computational geometry course taught in our department. Chapter 7 describes how this framework can complement and expand the educational experience that students receive in the course.

However some additional work is required to successfully integrate the system in the course structure. Among this work we can mention:

- document the framework and "publish" the documentation in public-accessible web site.
- expand the number of computational geometry demos based on the system. Design new coursework that could be solved by using the system.
- improve the framework robustness
- experiment with the system performance: find optimum sizes for student groups working concurrently in the system. Also determine the optimum number of servers required for the system in order to provide satisfactory response times for large number of users.
- explore and improve the scalability of the system. As a research system scalability was not one of Mocha design goals. However it can become an issue if the system is to be used by substantial numbers of users.

puting. Parallel algorithms can be difficult to design, understand and learn. An educational tool like Mocha implementing the animation of parallel algorithms for computational computing would be a powerful pedagogical tool. Among the parallel algorithms that could be implemented within the Mocha framework we can mention divide and conquer algorithms.

8.4 Support of 3D geometric algorithms

A natural extension for Mocha is the implementation of 3D algorithms. Some of the algorithms currently supported like convex hull and voronoi diagrams for example are well suited to a 3D implementation.

3D is not currently supported in Mocha because of present technical limitations of Java. On its actual incarnation Java supports only a 2D media API, part of its AWT GUI toolkit. There some existing 3D libraries for Java, like Liquid Reality,developed by Dimension X.

However these libraries are C-coded libraries with a Java “wrapper” around them. For this reason they are platform-dependent and cannot work from inside a web browser (without modification). These defeats the accessibility goal of Mocha and thus we decided against its use.

However new native 3D APIs for Java, currently in development will make possible support for 3D geometric algorithms, including “fly-through” and manipulation of animations of objects like 3D convex hull and 3-D voronoi diagrams.

Another option is the use of VRML 2.0, which will allow interactively and behavior supported by Java and JavaScript.

8.5 Shift to a component-based architecture

One of the current trends in the software industry is the development of component technologies. Components are building blocks that can be mix and match together to create complex distributed applications.

Components offer code reusability, since they can be used as “black boxes” without requiring a deep understanding of their inner workings. Object-oriented languages’ promise of greater code reuse remains to a great extent unfulfilled, because of the difficulty in designing object-oriented libraries that allow useful reuse of code.

Another benefit of components is that they can extend existing applications with new added functionality. By inserting a component on a standard application we can add the component functionality to the “container” application. The component can even “merge” its user interface with the interface of the application, by extending the menu hierarchy for example.

In the Internet world the main component technologies are Java Beans, which Sun is currently implementing for its Java language, and Active X, a component technology devel-

8.0 Future Work

8.1 Support of new collaboration frameworks and standards

The Internet is currently evolving in the direction of collaborative environments. Some of the major players in the software business including Microsoft and Netscape are positioning themselves to gain a foothold in the emerging groupware market for the internet and for corporate intranets.

Each of these companies have developed proprietary collaborative environments for their web browsers, namely Cool Talk by Netscape for the Navigator browser and NetMeeting by Microsoft for the Web Explorer browser. These environments typically include internet-based telephony, shared whiteboard, chat windows, and even sharing of standard applications (NetMeeting).

If these systems become open instead of proprietary and eventually converge into a established standard they would provide a useful framework for developing collaborative applications for academic research, like the Mocha system. In the case of Mocha we could take advantage of the internet support for example to provide audio conferencing in addition to the current text-based “chat” communication.

In addition Sun Microsystems has recently proposed a new set of extended APIs for the Java language including new support for 2D and 3D media, additional server support, telephony support, etc. If and when these API implemented the new capabilities of the language could be well exploited within the Mocha framework.

8.2 Support of new multimedia types

Collaborative Mocha can benefit greatly from supporting new multimedia types like video and audio. By supporting audio and video conferencing the system could improve the “awareness” provided by the multi-user environment, i.e. perception of other users in the system and their actions.

This extended multimedia support should be accomplished ideally by maintaining the Mocha philosophy of leveraging existing internet technologies and supporting and taking advantage of current standards. This approach eliminates the substantial effort required for implementing proprietary solutions.

This support could be implemented by adopting one of the alternative collaborative systems discussed in 8.1. Other possible options include the use of the proposed Java Multimedia extensions by SGI (new 2D Media API) and others (SGI Cosmo, etc.).

8.3 Animation of Parallel Algorithms

The work done so far with Mocha and Collaborative Mocha has been limited to serial algorithms. However a next step worthy of consideration lies in the field of parallel com-

course and the TA's. The virtual meetings would offer a convenient way for asking questions and exploring further the concepts aided by on-line support from the instructor.

This virtual course could be offered under the auspices of the Center for Geometric Computing and can be developed as a joint project between Brown University and the other universities involved in the Center.

The assignments of the course could be done remotely by the students. The completed projects could be either emailed to the instructor or TA's, published in a web page, or sent over by more traditional means like regular mail.

7.3 the Framework as an educational experience

The Mocha framework provides a rich set of classes encompassing geometry objects, user interface widgets, communication support objects, geometry algorithms, etc.

Traditionally students in CS252 are given the liberty to choose the language and libraries to implement their projects. However they often have to rely on general-purpose libraries that require considerable effort to create the support code for the application, including graphics, user interface, data structures, etc.

By providing ready to use classes designed and tailored specifically for computational geometry we can reduce substantially the amount of time necessary to create the support code for implementing geometry algorithm. Thus the programmer can concentrate on implementing the actual algorithms.

Ideally students should learn algorithms by programming them. In a computational geometry course time limitations can preclude having the students program a large number of the algorithms taught in class. However the use of a framework like Mocha allows shifting the emphasis of the course from the realm of the strictly (or mostly) theoretical into a more balanced mix of theory and implementation.

sary infrastructure the students could concentrate on programming the actual algorithms, without having to spend time creating the required geometric objects or UI widgets.

The use of this framework as integral part of the course could produce additional interactive demos for future use as didactic tools, and would make programming a more substantial part of the course, improving and expanding the learning experience of the students in the course.

For this purpose the course could require periodic programming assignments, in addition to the final project currently required of the students. The use of the framework should reduce the effort and time in the part of the students to implement the algorithms, so ideally this assignments should not take an excessive amount of the time the students spend on coursework.

7.2.2 learning-at-a-distance: Mocha as a self-pace study aid for individual, remote learning.

Face to face collaboration and learning is not always possible. Sometimes it is necessary (and maybe even desirable) to study a subject individually without the benefit of a classroom or a instructor. The World Wide Web has proved to be an excellent medium for distributing information and knowledge. The multimedia capabilities of the Web provide a powerful setting for an learning experience and environment.

An example of a Web Site created to distribute academic information, is the Brown CS department home page (http://www.cs.brown.edu), and more specifically the home page for the CS252 Computational Geometry course (http://www.cs.brown.edu/course/cs252). The home page for the course contain links to documents of all the class materials, interactive demos built by previous students of the course, hyper-link references to related material, etc. that students (and other people interested) can study at their own pace.

Mocha can serve as an programming framework for creating interactive illustrations for the Web. To demonstrate this potential we have created an interactive, on-line version of the original paper that introduced the Mocha architecture. This paper combines hyper-text references, model animations, interactive illustrations and explanatory text. This document is located at the URL http://loki.cs.brown.edu:8080/

The Mocha framework documentation is available on-line, and the source files for the framework could eventually be made public in order to offer the opportunity to others outside Brown in the Computational Geometry community to take advantage of the benefits of the framework.

7.2.3 Cooperation-at-a-distance: Mocha as a virtual classroom.

Another alternative is to use Mocha as a medium for creating a virtual Computational Geometry course. The CS252 home page contains already the text of the classes given in the course. Weekly virtual meetings similar to regular classes could be scheduled to give remote students the opportunity to interact with each other and with the instructor of the

- Use of multifaceted learning tasks for cooperative group investigations.

The teacher identifies a specific task. The student teams work together to plan and carry out the task.

- Inclusion of multilateral communication among students and encouragement of active learning skills.

The different teams prepare a report to present to their class for discussion and evaluation. People in the same team work closely together, but there is usually little cross-team interaction.

- Teacher exchange with each of the groups.

The teacher works with the individual teams, providing feedback on their work and giving directions as needed. The Teams report back to the teacher and to the class as a whole.

7.2 Scenarios for Mocha

We have considered uses for Mocha both in a co-present environment and as a cooperation-at-a-distance tool. In this section we will describe how Mocha can provide educational benefits in these two situations.

7.2.1 Co-present environment: Mocha as an educational aid for a traditional computational geometry course (i.e. Brown's CS252)

We propose to extend the traditional theoretical classes in CS252 with a weekly session in Brown's electronic classroom (also know as the sunlab). For these weekly sessions the class could be divided into sections. Each section would be coordinated by a Teaching Assistant (TA). Sections would have their own schedule and meeting times (to reduce congestion of the electronic classroom).

In these sessions the students could use interactive multimedia demos of the theoretical concepts seen in class that week. If case of any doubts they could ask questions to the TA's or to other members of the section. In addition each week a problem set or a task could be offered to the section, and the students would work cooperatively to solve the problem set or achieve the given task. This work should be complementary to the concepts taught in the theory classes.

This approach is similar to the scheme used in some of Brown University's CS department courses like CS141 and builds on the experience gathered in building educational activities in the electronic classroom.

The electronic classrooms meetings would also include "help" sessions where the TA's could teach the basics of the Mocha framework. This framework could then be used by the students (either individually or in teams) to implement computational geometry algorithms taught in class during the course. Since the framework provides much of the neces-

7.0 Mocha in the Classroom

Collaborative learning or “group” learning is designed to encourage cooperation, the discussion of ideas, resolution of cognitive conflicts, and promote problem solving and higher-order thinking skills [9].

The advantages of using cooperative learning for the education of theoretical fields like mathematics has been explored and stated by researchers like Davidson []. Group learning can address some of the problems typically associated with the potentially isolating nature of math-related courses.

It is generally recognized that students can become less discouraged when working in a group as compared to students working alone. The group provides not only a source for additional help, but also becomes a support network for its members.

By combining computers with group learning in math and geometry-related fields in a Computer Supported Collaborative Learning (CSCL) environment we can empower the students to construct and explore mathematical and geometrical objects and worlds.

Collaborative Mocha is an attempt to bring the benefits of CSCL into the field of Computational Geometry. In this section we will propose the use of Mocha and Collaborative Mocha as an educational aid for the teaching of computational geometry.

Mocha as a tool is not meant to replace traditional classroom instruction. We envision Mocha as a complement to traditional classes. However it can also offer benefits in a self-paced individual study setting. Next we will discuss how to organize a classroom in order to take full advantage of the system, and propose some scenarios describing both realistic uses of Mocha and its educational benefits.

7.1 Group Investigations within classrooms

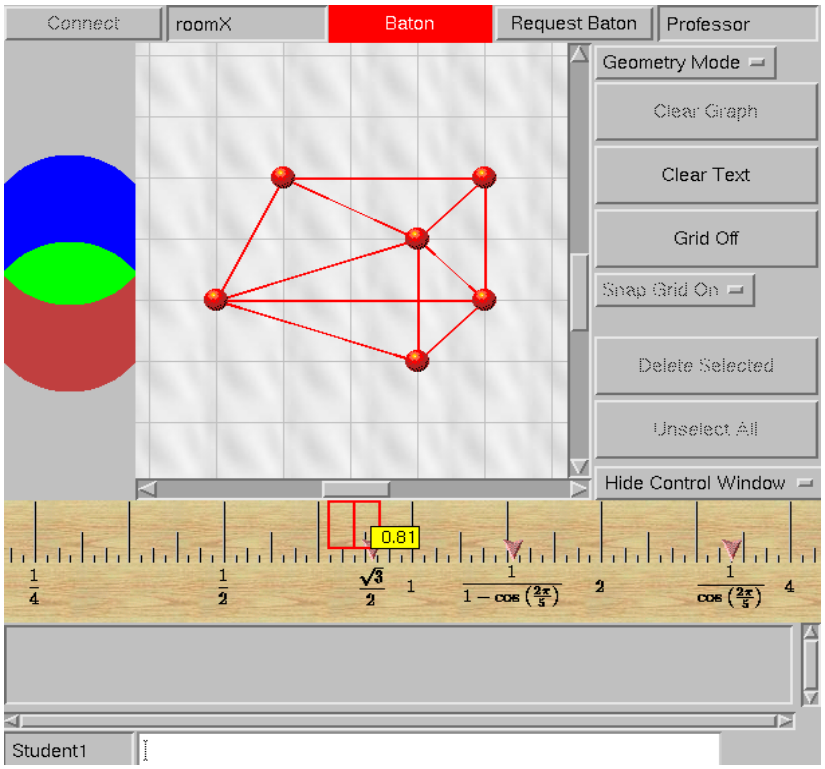
Group learning usually requires some reorganization in the classroom. Typical classrooms have a large number of students that often makes it impractical to arrange collaborative activities without organizing the class into smaller teams. In addition group learning redefines the role of the teacher, from a mostly unidirectional exchange to a fully bidirectional experience.

By having the teacher interact with teams instead of having to interact individually with each and every student we can keep his work more manageable Investigators like Sharan and Hertz-Lazarowitz have developed procedures for organizing group investigations.

Successful organization of group learning requires the coordination of four dimensions of classroom life:

- The organization of the classroom into a “group of groups”

The class is divided in small groups to form teams. Team members work cooperatively. Inter-team work can be cooperative or competitive.



6.5 Persistence and History

Each room exists in the server until it is explicitly deleted. If the user leaves a room and comes back later, it will still be there, even if there is no one in the room. The user can also explicitly save or restore the room state to/from disk. Persistence in Mocha exist in both primary and secondary storage.

Rooms persist in memory unless they are explicitly removed by the user. In addition they can be stored in disk, and recalled at will. It is important to note also that persistence exists only in the server side, not on the client. Java applets have security restrictions imposed by their very nature, as applications “downloaded” on demand by a remote server whose security cannot be guaranteed.

In order to reduce the risk of a possible destructive action by a Java applet the Java Virtual Machine restrict access to certain local resources from the applet, including access to the local filesystem. This limitation makes impossible the definition of local persistence. Persistence is achieved by maintaining an individual and independent history for each room. The history has three components:

- the geometry state, with a current image of the geometry model in the room.
- the sketch history: with a list of all sketching commands issued by users in the given room
- the chat history: a list of all the chat text sent over by users in the room (identified by originator).

6.4 Communication and Personal Interaction

6.4.1 Conversation

The whiteboard allows different levels of conversation between the various users in a shared room. On one level they all have access to the geometric model (by using the baton) and can manipulate it jointly, in a common visual interaction. Other mechanisms provided are the chat Panel, which allows text-based conversation, and sketching, which allows visual gestures. The overall “conversation” between the users of a given room is the sum of these schemes.

6.4.2 Gesturing

Gestures are a non-verbal form of communication usually involving body language. Since the CM system doesn’t currently support videoconferencing, the users of the system can’t see each other, so visual body language is out of the question. Sketching is offered as a alternative way of visually expressing non-verbal gestures to remote users.

This form of gesturing is supported by sketching and annotation capabilities over the background. this allows the user to pinpoint particular features in the geometric model, or the whiteboard in general, and permits a greater degree of communication. The sketching options are quite flexible and lets the users define their own schemes (like different shapes or colors) if desired for identifying the owner of the gestures. Gesturing is available to all members of the room at all times, regardless of baton possession.

6.4.3 Data Sharing: locking mechanisms

A collaborative system can very easily degrade into anarchy if no forms of coordination are provided. If users try to interact simultaneously with the same object, for example by moving the same node in opposite directions, the result will be that neither of them will succeed on his interaction.

Mocha provides a means for coordinating and arbitrating between multi-user interaction based on a “pass the baton” control model. Each room has a single baton. When an user owns the baton he gains exclusive control over the room geometry model until he yields the baton.

An user can request the baton at any time. If there is no current baton owner the baton is granted automatically by the Server. If there is already a baton owner the request is transmitted to him. The baton owner can then yield the baton or reject the request. If the baton owner leaves the room the baton is declared free.

This allows organized interaction between multiple users sharing the same working space. In the screen image above we can see 2 nodes enclosed in a green box, which have been “selected” by the baton owner. The baton indicator on the top session information panel is green, indicating possession. On the right of the session information panel we can see the name of the baton owner.

We use a color scheme reminiscent of traffic lights (which should come naturally to most people). The color green is used to indicate that an asset is controlled by oneself. The color red indicates lack of control. This color scheme is used with the baton possession indicator on the session information panel. Any geometry model elements selected by the baton owner are also shown bounded in green.

The visual cues work in conjuction with an internal locking mechanism that keeps collaborating users from having interaction conflicts in the shared environment.

6.3 Synchronization

Mocha provides a synchronized What You See Is What I See (WYSIWIS) shared environment. Collaboration systems range from the completely asynchronous to the fully synchronized. Under WYSIWIS users see the same objects, in the same ways and the same places, and it has been proposed as a foundation concept for shared systems [Stefik et al].

WYSIWIS requires a tightly coupled synchronization between multi-user interfaces. However a strict WYSIWIS implementation requires certain overhead necessary to keep all the user environments synchronized. Any change introduced by one user has to be reflected instantly in the environment of the other users. This overhead is reflected in the need of complex locking mechanisms and heavyweight and costly network protocols.

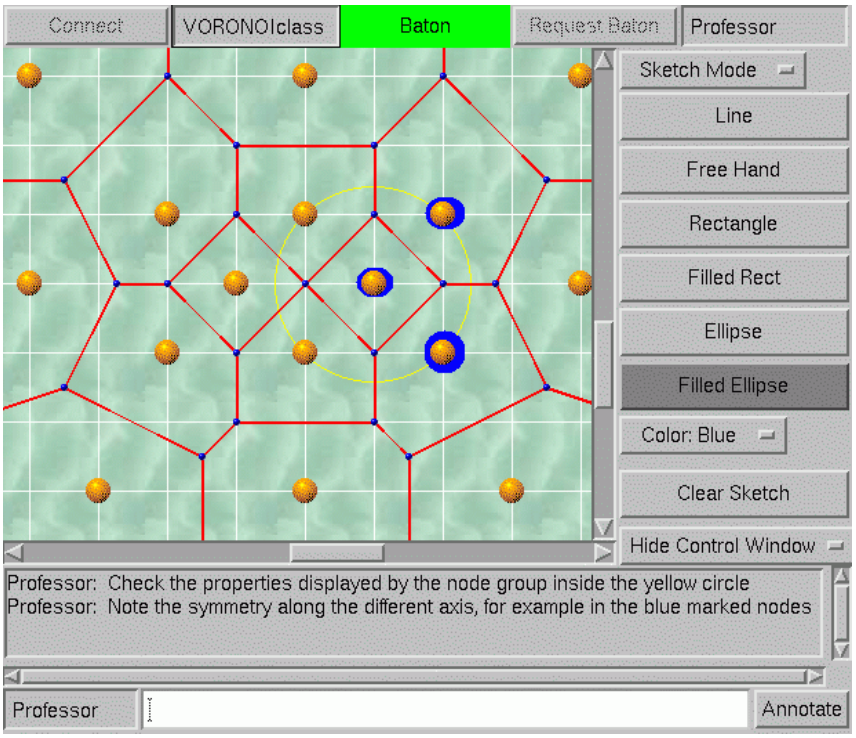
“Relaxed” WYSIWIS is a related form of synchronization proposed by [Stefik et al]. It allows synchronization constraints to be relaxed along four key dimensions: space, time, population and congruence. Mocha supports some of the relaxed WYSIWIS features, but it is more strict in other aspects.

- space: WYSIWIS is applied only to a subset of visible objects. In Mocha objects inside a room are always synchronized (event if not visible) to users of the room.
- time: allows delays in updating views. Mocha users who simultaneously share a room are always synchronized (with minimum delays).
- population: sharing is limited to subgroups of the user population. Mocha for example synchronizes only users sharing a common “room” or session, not all the users present in the system.
- congruence: allows alternative views of objects. Mocha supports alternative views of the geometric model, possible by the MVC model on which the system is based. However the prototype system that we have in place is not yet taking advantage of this feature.

The persistence qualities of Mocha rooms support a form of asynchronous collaboration. Users can access a common room at different times and examine the changes that the other introduced in the room. However simultaneous use of a shared room is always synchronized.

This canvas is customized on each of the Mocha clients to support the different geometric services provided by the Mocha servers.

In the following screen image we can see an example of a Voronoi diagram shared canvas, where the user is using sketching in order to hilight some features in the diagram, and using the chat panel to communicate with fellow users in the shared canvas.



6.2.2 Conversational text

Mocha supports a “chat” panel widget where all the collaborators in the room can “talk” to one another via text. The chat panel will identify every line of text by its author name. This chat widget supports a form of written communication that complements the

6.2.3 Sketching

The system allows “sketching” and annotation over the background in order to provide additional support for gestures. However the geometric model is always on the foreground of the whiteboard.

6.2.4 Visual feedback and responsiveness

All interactions (either local or remote) generate immediate (within reason, depends on network lag) visual feedback. Visual “cues” indicate who has control over the whiteboard.

the user can keep track of the different rooms available on the server and the users that are in a given room. The user can go very quickly and without restrictions from one room to another. In the Control Window the user can also define his name, or change it.

In case of a change of name the new name will be updated in all the other clients sharing the same room. If the user leaves a room all selected geometry objects will become unselected and all other clients will be updated as well.

The control window has on the left side a list widget that contains the rooms compatible with the client type that exists on the environment at the given time. On the right side of the window it is located another list widget that contains all the users sharing the same room with the local user. On the center there is a button panel widget with different activities that the user can do.

The activities include creating a new room, removing a room, saving the room contents to disk and restoring the room content from disk. In order to remove a room the user must be the only one left in the room (he must be in it at the time). In order to select a room the user double clicks on the room's name in the room list widget. Every room is identified by a unique name.

In the bottom of the window a text label indicates the room where the user is. In the top of the window a text label describes the name of the user. If the user changes his name (by using the change name button) a input field appears for that purpose.

The Session Information Panel on the top of every geometry client provides information about the current room, current baton owner in the room, a baton possession indicator, and a baton request. Every user in a room knows at every time who the owner of the baton is. Every action on the geometry model is clearly identified by the baton owner.

6.2 Visualization

Mocha supports different forms of multimedia information. This information is presented to the user in different ways to allow different degrees of visualization. Among the information handled by the system we can mention geometry objects, conversational text, and "sketches" or drawing information. The user can interact with these different types of data according to their nature. The geometric objects can be manipulated to change their attributes: the user can have geometric algorithms run over them and see the results.

6.2.1 Geometric model

The geometric model varies depending on the type of client, but usually will include geometric objects like graphs, nodes edges etc. We support a MVC model where the underlying geometric data could potentially be displayed in different ways depending on the particular geometric client used.

In order to provide a space for displaying the geometric model and have the users interact and collaborate with the geometric objects the Mocha clients provide a geometric canvas.

Rooms were proposed originally as a single-user application mechanism for organizing collections of windows in related screenfulls of information[Henderson]. Other researchers proposed extending this idea to multiuser situations [Stefik et al].

The use of rooms as an underlying metaphor for a collaborative system is not unique to Mocha. As an example of another system that makes use of the room metaphor we can mention Mushroom [Kindberg], a framework for collaboration and interaction across the Internet.

Each CM room is completely independent of one another, has it is own unique state and history. Since rooms are persistent, users can interact either synchronously (in the same room at the same time) or asynchronously (users occupy the room at different times, but observe each other's changes to the shared data).

There is no physical limits on the number of people on each room or the number of rooms on the server. (of course, there are practical limitations imposed by external factors like network latency). Rooms can be created and removed dynamically by the user.

Each room has a "type", according to the activities that can be carried out in that room. The supported activities correspond to the geometry and animation services provided by the Mocha server. The rooms of a given type will be accessible only to Mocha clients who support that type. The server itself supports all client types and all activities.

Rooms are accessed by pointing the web browser to the Mocha URL. After the Mocha client starts it will request and receive from the server a list of all the existing rooms of the same type as the client. The user can either choose an existing room or create a new one.

After the user selects a room the client will receive and display a list of all the other users in the room. This list will be update automatically every time a user enters or leaves the room. Users can go from one room to another by simply selecting another room.

Each room has a single "baton". The baton gives its owner sole control over the geometry model in the room. The sketch or chat text capabilities of the room require no baton and can be accessed by any room member at any time.

6.1.2 Explicit Embodiment

Every user has a name that identifies him to the other users in the same shared working space. This name represents the identity of the user in the Mocha environment. The name will identify both the presence of the user and some of his actions to other users.

The position of the user in the environment is defined by the room where the user is at the moment. The presence of a user is shown to other users only in the context of a shared room. Users are aware only of the presence (and thus the position) of the users who are in the same room at a given point. This allows a degree of privacy to users.

Mocha conveys explicitly identity and presence of the users by means of a Control Window and a Session Information panel. Every Mocha client has a Control Window where

6.0 Collaboration in Mocha

6.1 Awareness and embodiment

One of the important issues that raises in collaborative environments is the one of awareness and embodiment. Awareness is the perception that a user of a collaborative environment has of the presence other users sharing this common environment. It is also related to how the users perceive their own and each other actions.

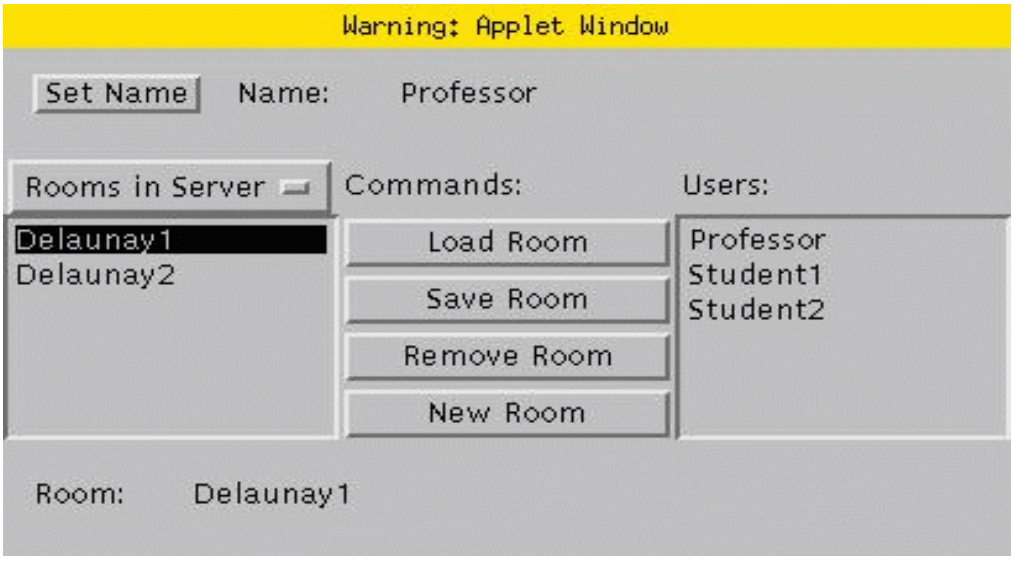
Collaboration requires developing flexible mechanisms so that users can manage their awareness of other users and so they can encourage other's awareness of them. A related issue is embodiment, how users are going to be represented to one another in relation to the information they are using.

Among the issues raised in awareness we can include conveying presence, position, identity, activity, gestures and "verbal" communication (voice or text based), etc.

6.1.1 Room Metaphor

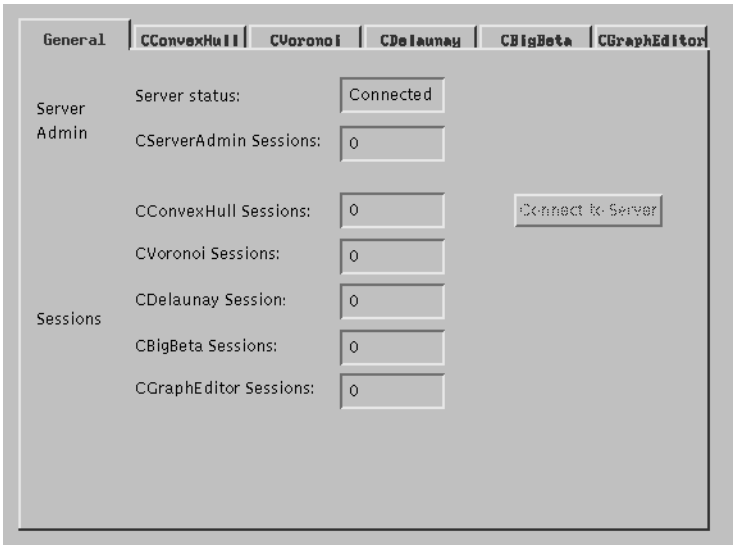
The internet is spread world-wide across traditional boundaries like geography or nation-states. Since Mocha uses the internet and the WWW as its distribution channel the system users can be located anywhere in the world. Some sort of collaboration infrastructure is required to enable remote user interactions and collaboration based around mutual goals and shared data.

Collaborative Mocha relies on a "room" metaphor in order to organize the interaction of different groups of users (both remote and local). In common speech a room is a physical location where people gather for some purpose, for example learning in a classroom. Mocha virtual rooms are environments for collaboration and containers for shared data.

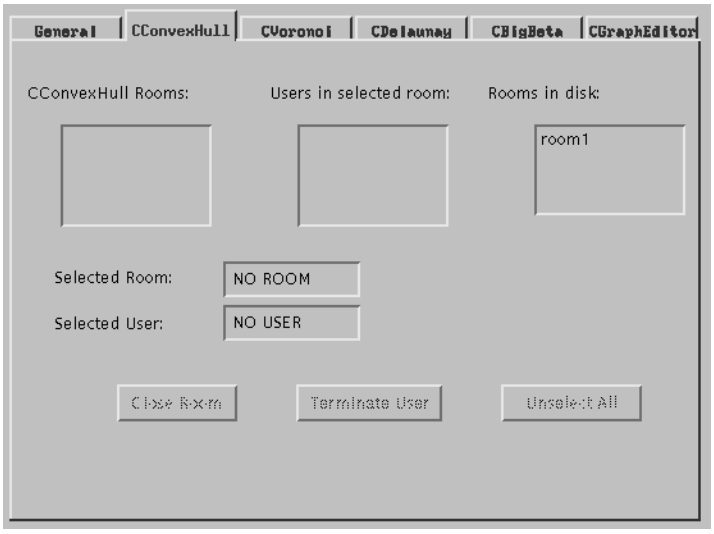


tor has the option of terminating any of the users if he desires to do so. Any active room can also be closed by the administrator. In addition a list of all rooms on disk of that particular type is shown to the user.

The administration client is a tool for both visualizing the state of the server at any time and for managing the server. This client does not implement a security mechanism at this time to restrict the access of non-authorized users. However such a security mechanism would be a desirable addition in the future given the level of control granted to this client.



General “Tab” Panel of the CServerAdmin client



CConvexHull “Tab” Panel of the CServerAdmin client

Each algorithm produces a different visual result as a product of running the algorithm. For example the search algorithm results will be represented in the client as “visit numbers” attached to the nodes, indicating the order in which the nodes where visited in the search. In addition the visited nodes will be marked by a yellow filled circle.

The connected components algorithm results are represented by color-filled circles surrounding all the nodes in the geometry canvas. Each color represents a different connected component. The user can identify visually right away which nodes belong to which component. Each edge is colored as it is considered by the algorithm.

In the Minimum-Spanning-Tree algorithm by Kruskal the orange color is used to indicate that an edge is part of the Minimum-Spanning-Tree. In addition the same color scheme is used to represent the different connected components to which the nodes in the graph belong. As the algorithm progresses the number of connected components present in the graph will diminish as the Minimum-Spanning-Tree is built.

The CGraphEditor client uses a specific Animation Panel GUI component to let the user control the algorithm animation. This panel defines a Algorithm choice widget for selecting the algorithm, and a button tool bar with functions like start, for starting the algorithm, step ahead and step forward, for running the algorithm one step a a time, and clear, for clearing the visual results of the algorithm from the geometry canvas.

The different algorithms are run in the Collaborative Server, in accord with the architecture of Collaborative Mocha. The Model Manager object in the server has been extended to support the additional data structures needed for each of the animated algorithms. The algorithms themselves are coded into the server and when run send update events to the different clients sharing the graph editor room where the algorithm is being executed.

The algorithm animation is kept synchronized between all the involved graph editor clients keeping the WYSIWIS philosophy adapted by CMocha. Any changes on the graphs created and edited in the CGraphEditor is immediately reflected on both the Collaborative Server and all CGraphEditors sharing the same room.

- Collaborative Server Administration Tool

The collaboration administration tool (CServerAdmin) lets an user administrate the Collaboration server. This tool has several “tab” panels, one for each type of geometry clients that the CMocha server currently supports, and one main panel for general information about the operation of the server. The user can selected which “tab” panel to display by clicking on the tab name (on the top of the panel).

This main panel has information about the number of total connections the server has for each type of client and the status of the server.

The client-type panels display the number of active rooms that the server has of that particular client, and given a particular room, what users are in it at the time. The administra-

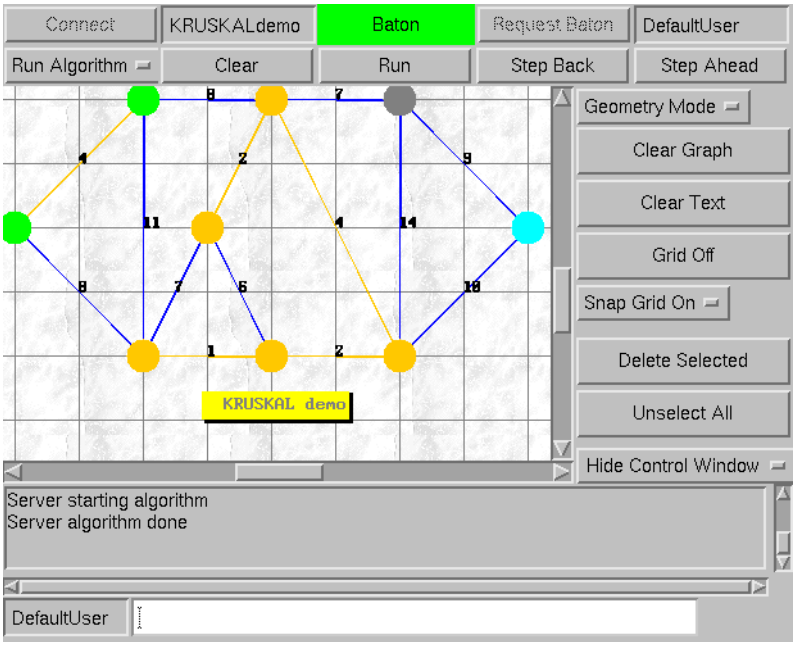
- New collaboration-only clients: Collaborative Graph Editor with animated graph algorithms (CGraphEditor)

This new client defines a collaborative graph editor which allows the user to input and manipulate a graph. The user can interactively create and manipulate nodes and edges. The graph can be either directed or undirected, and the edges can have weights attached to them.

This collaborative graph editor has a complete UI for entering and manipulating graphs and for choosing, configuring and running algorithm animations over them.

The user can then execute an animated algorithm over the graph and see step-by-step how the algorithms runs and arrives to the result. The algorithm can be executed and animated with both forward steps and back steps. The algorithm currently runs on the CM server, and the client only displays the results received from the server.

Six graph algorithms have been implemented so far: Breadth-First Search (BFS), Depth-First Search (DFS), Connected Components, Minimum-Spanning-Tree Algorithms by Kruskal and Prim, and a Minimum-Path Algorithm by Dijkstra. However many other algorithms can be implemented using the infrastructure provided by this Graph Editor client and the CMocha collaborative Server.



Depending on the algorithm specified by the user additional information can be entered about the graph. For example if one of the search algorithms is chosen then a root node can be selected. If the selected algorithm requires it then a weight can be entered for each edge in the graph. The additional information will be hidden from the user if it is not necessary for the currently selected algorithm.

- Color Node: a graph node is drawn with a specific color to indicate some attribute that the running graph algorithm has changed. For example the color may indicate that the node was visited in a Bread-First-Search, the connected-component to which the node belongs (in a Connected-Components algorithm) etc.
- Uncolor Node: reverts the node to its original representation (usually a shaded 3D sphere image in CMocha) indicating that no attributes has been changed by a running algorithm.
- Color Edge: a graph edge is drawn with a color to indicate some attribute that the running graph algorithm has changed. For example the edge can be part of the result tree in a Minimum-Span-Tree algorithm like Prim or Kruskal.
- Uncolor Edge: a graph edge reverts to its original representation (a line with a default color) indicating that no attributes has been changed.

5.5 Extensions in the Framework

The framework has been extended to support the new features added to the collaborative environment. The geometry classes of the framework which include points and pointsets, graphs, nodes and edges etc. have been extended to support new attributes. The selection and locking mechanisms for example (discussed in the next session) require a multi-value "status" attribute in each selectable geometric object.

Other clients implemented like the graph/editor with animated undirected-graph algorithms require new attributes as well particular to the animated algorithms, for example a "visited" attribute on the graph nodes, or a "cost" attribute in the graph edges. The new attributes require a standard representation in the protocol, a standard data storage form and a graphical representation for use in the GUI.

In addition new widgets have been added to support the GUI extensions mentioned before, and the communication-support classes provided by the framework have been extended to support the extensions to the protocol.

5.5.1 New Collaborative Clients

For an usage description of Collaborative Mocha refer to the user's guide [Lejter 1997].

- Collaborative versions of previous clients

As a proof of concept for CM we have implemented collaborative versions of the original NC Mocha clients developed. They include CConvexHull (Convex Hull diagrams), CVoronoi (Voronoi regions), CDelaunay (Delaunay triangulation) and CBigbeta (Big Beta proximity graph). By an adopted convention the preceding C in the name indicates that the Mocha client supports collaboration.

This collaborative versions support all the geometry functionality as the original versions but have in addition all the added collaboration functionality defined in the CMocha model by means of the extensions in Protocol, GUI, and Framework described above.

5.4.2 New Session management protocol

CM adds a new session management protocol. This protocol supports all the session management commands that the user can access at any time. They include adding a new session, removing an existing session, saving or restoring a session to/from disk, requesting (and receiving) the list of existing sessions in the server, requesting (and receiving) the list of users in a given session, requesting a change of name for a given user, clearing the history of a given session (either sketch or chat history), clearing the session canvas, or updating the session state for a given client.

TABLE 2.

Command	Description
ct	CMD_TEXT
cs	CMD_SKETCH
cls	CMD_LIST_SESSIONS
ccst	CMD_CLEAR_SESSION_TEXT
ccb	CMD_CLEAR_BACKGROUND
cas	CMD_ADD_SESSION
cus	CMD_UPDATE_SESSION
crs	CMD_REMOVE_SESSION
css	CMD_SELEVT_SESSION
csts	CMD_SET_TYPE_SESSION
clvs	CMD_LEAVE_SESSION
clse	CMD_LOAD_SESSION
csse	CMD_SAVE_SESSION
ccn	CMD_CHANGE_NAME
cq	CMD_QUIT

The session protocol also supports the communication of chat and sketching information between clients. This information is not processed by the server, only stored in the history of the respective session and rebroadcast to “peers” or clients on the same session. There is no peer-to-peer direct communication in CM. The server is always accesed first and it turn it will broadcast information as needed to other clients.

Other session protocol messages support “baton” requests or replies between users of a shared session. The baton owner commands sole control over the geometry model.

5.4.3 New Animation Events

Four new animation events have been defined to support the graph algorithm animation services provided by the CMocha Server (see section 5.2.1) and the new Collaborative Graph Editor described next in section 5.5.1.

These events are:

5.4 Changes in the Protocol

5.4.1 Atomic operations in the animation protocol

The animation protocol of Mocha is based on the LEDA file format. In NC Mocha any change in a geometric object required transmitting the whole object over the network to the server even if the change was limited to a minor attribute in the object. This resulted from the fact that the NC protocol doesn't support atomic operations, and reduced the efficiency of the protocol.

In the NC system the reduced efficiency is not a significant problem. In the CM version the additional overhead of synchronizing multiple users sharing the system produces significant lag and delays. In order to diminish the network lag and improve the provenance the CM protocol has been extended with atomic operations. The atomic operations include selecting a subset of nodes, moving a subset of nodes, removing a subset of nodes, marking a node as visited, etc.

TABLE 1.

Command	Description	Atomic Operation
ag	ALG_GRAPH	NO
asr	ALG_SET_ROOT	YES
asd	ALG_SET_DELAY	YES
asn	ALG_SEL_NODE	YES
avn	ALG_VISIT_NODE	YES
auvn	ALG_UNVISIT_NODE	YES
acv	ALG_CLEAR_VISITED	NO
aan	ALG_ADD_NODE	YES
arn	ALG_REMOVE_NODE	YES
amn	ALG_MOVE_NODE	YES
amsn	ALG_MOVE_SELECTED_NODES	YES
asb	ALG_SET_BETA	YES
aps	ALG_POINT_SET	NO
acps	ALG_CLEAR_POINT_SET	NO
ach	ALG_CONVEXHULL	NO
av	ALG_VORONOI	NO
ad	ALG_DELAUNAY	NO
ab	ALG_BIGBETA	NO
asa	ALG_START_ALGORITHM	NO
afa	ALG_FORWARD_ALGORITHM	YES
ara	ALG_REWIND_ALGORITHM	YES

of the canvas shown in a given moment. In addition we have added the feature of snap-to-grid node creation to simplify the positioning of nodes along specific lines.

When dragging a selected geometry element in the canvas the canvas will scroll automatically if the element is about to go off the show area. If the actual edge of the virtual canvas is reached the element will stop at the edge.

Different users can see different areas of the common virtual canvas at any time. However any change introduced by one of the users will be reflected in all the other users' screens (who are sharing the same session).

- Sketch Tool Panel

The sketch tool panel adds drawing capabilities to the geometry canvas. They include lines, rectangles, filled rectangles, ellipses, filled ellipses with the ability to choose color. The sketching is provided as a resource for "gesturing", or expressing ideas in a non-verbal way. The client tool panel can be switched between sketch and geometry mode and will show the appropriate tools accordingly. When the client is on sketch mode the user can't interact with the geometry objects in the canvas.

- Geometry Tool Panel

The geometry tool panel provides widgets for selecting different activities that the user can perform with the geometry canvas. They include selecting multiple nodes or edges, moving selected nodes, deleting the selected nodes and edges or unselecting all. The canvas grid and the snap-to-grid feature can also be turned on or off.

- Chat panel

The chat panel offers widgets for "verbal" communication between users. Since they system currently doesn't support audio-conferencing the conversation is written instead of spoken. The chat panel includes a text area widget which contains the text sent by the different users (each lines is identified by the user name) and a input field which is used to submit new contributions to the text conversation.

notion of session persistence and gives the Session Manager the ability to save and load sessions from/to disk.

5.3 Changes in the Client

The client in CM has been extended both in the GUI and the animator components to support collaboration.

5.3.1 Animator's extensions

The animator now includes two threads running in the background in addition to the main program thread which handles the user interaction. One thread (present in the NC version of the system) updates the display at regular intervals.

An additional thread is dedicated to reading data from the socket connection to the server. In NC Mocha all interactions between the server and the client where synchronized, using a request-reply scheme. CM introduces new asynchronous interactions between server and clients. If more than one user is working in a given session, every change in the session's state will be broadcast to all users sharing the session. The update message will be a asynchronous interaction between the server and the users who didn't originate the change.

The CM protocol supports also atomic operations. If a change is introduced in the geometry model for example it is not necessary to transmit the whole model, only the element or attribute that changed. This feature supports incremental communication between the clients and the server (and viceversa). The use of a separate thread to read from the client socket also guarantees that the client won't block by trying to read from the empty socket.

5.3.2 GUI's extensions

- Control Window

The control window offers a convenient widget for managing the session services that CM offers. The control window can be turned on or off (hidden or shown again) so it doesn't get in the way when the user is not using it.

- Session Information Panel

The session information panel placed at the top of the client's main window displays the current room, the baton owner's name, a baton request button and a baton indicator.

- Scrollable geometry canvas with snap-on grid

NC Mocha's geometry canvas offered a limited working area restricted in size. While this limitation didn't impede useful work in single user mode it was considered too inflexible to allow useful collaboration in a multi-user environment. CM introduces a virtual working space larger the actual window containing the geometry client. This is achieved by using a scrollable geometry canvas with horizontal and vertical sliders to control the area

The connection manager interacts closely with the session manager. The connection manager contains all the connections that the server keeps at any given moment. The connection manager processes the raw data read from the socket by the connection and parses it according to the animation and session protocols. The parsed input data is then submitted to a geometry manager which runs the requested service over it. The result is then broadcast by the connection manager to all the connections sharing the session who originated the request.

The connection manager keeps polling every connection to determine when data has been sent over from a client and needs to be read and process. This approach avoids blocking the server by trying to read from an empty socket. By using polling instead of multithreading the server design is simplified and made easier to maintain.

- New Model and Services Managers

The geometry model manager provides the geometry model state for each of the sessions in the server. It partners with the services manager to provide geometry and algorithm animation services to the clients. The services manager provides a higher level, common interface to the LEDA library geometry algorithms supported by the Server.

The Model Manager now supports the animation of graph algorithms. The Model Manager provides some standard data structures for implementing them like a queue than can be used for FIFO, LIFO or keyed operation, and matrix storage for building an history of the algorithm run to be used in stepping backward or ahead in the algorithm execution.

Four standard methods are provided for implementing the algorithms: startAlgorithm() executes the algorithm on its entirety and broadcasts each step; forward algorithm() executes and broadcasts a single step ahead; rewindAlgorithm() executes and broadcasts a single step backward; runAlgorithm() executes the algorithm on its entirety but doesn't broadcast the steps or result. It is used to store the history of the algorithm execution to be used by forwardAlgorithm and rewindAlgorithm().

The Model Manager keeps an enumerated list with the algorithms supported by the Server. When each of the four algorithm methods it is called, it checks the selected algorithm type and calls the correct method for the algorithm selected. In order to extend the Server to support additional algorithms it is only necessary to add the desired type to the enumerated list, and implement the appropriate methods.

Both managers encapsulates all the geometry data and functionality provided by the server simplifying the work of the programmer maintaining or extending the system. The model manager works together with the connection manager. The animation protocol is only supported by the connection manager who does all necessary mapping to the data structures used by the model manager.

- Persistence

NC Mocha has no notion of persistence. The geometry state exists in the server only a the given time. If the user leaves his session, the state is gone forever. CMocha introduces the

CM discards NC Mocha's server based on external components and replaces it with an integrated server that provides both multi-user session support and geometry services. The integrated server scheme is more flexible and more suited for providing shared state between multiple users.

CM's sessions are multi-user shared working spaces. Each session has a collection of socket connections between the integrated server and the clients working on the common working space provided by the session. The session manager contains a collection of sessions and provides session management services.

5.2.1 Integrated Geometry Server

- Connection object

A connection in the CM server corresponds to a socket connection between the server and a client. The connection has a type which depends on the client type. It is either associated with a session of the same type or it is on "standby" mode. when the connection is on this mode the client can only request session management commands like creating a new session or selecting an existing one.

- Session object

CM's sessions have several components.The first component of a session is state. The state kept by the session includes a geometry model (which depends on the session type), a chat history, and a sketching history. Each session has its own independent state, which is shared among the member clients sharing the session. This state is persistent over time and can be save and restore from permanent storage like disk.

The second component of the session is a connection collection. This collections holds the connections with the clients sharing the given session. The connections are synchronized. Each client keeps a copy of the session state. When a change is introduced in the state by user interaction or algorithm result the change is "broadcast" by the server to the other connections of the session in order to keep the copies of the state synchronized. The session object has also a single "baton owner" at any given time, who has sole control over the session geometry model.

- Changes in the session manager

The session manager keeps the collection of sessions available in the server at a given time. The session manager also provides session management services to clients. These services include creating new sessions, selecting a session, requesting a list of users from a given session, saving or restoring a session to/from disk, etc.

In order to support the session management services provided by the session manager CM adds a new session management protocol. Both protocols are supported by the new connection manager. Any session services request parsed by the connection manager is passed over to the session manager.

- New connection Manager

5.0 Architecture Extensions for Collaboration

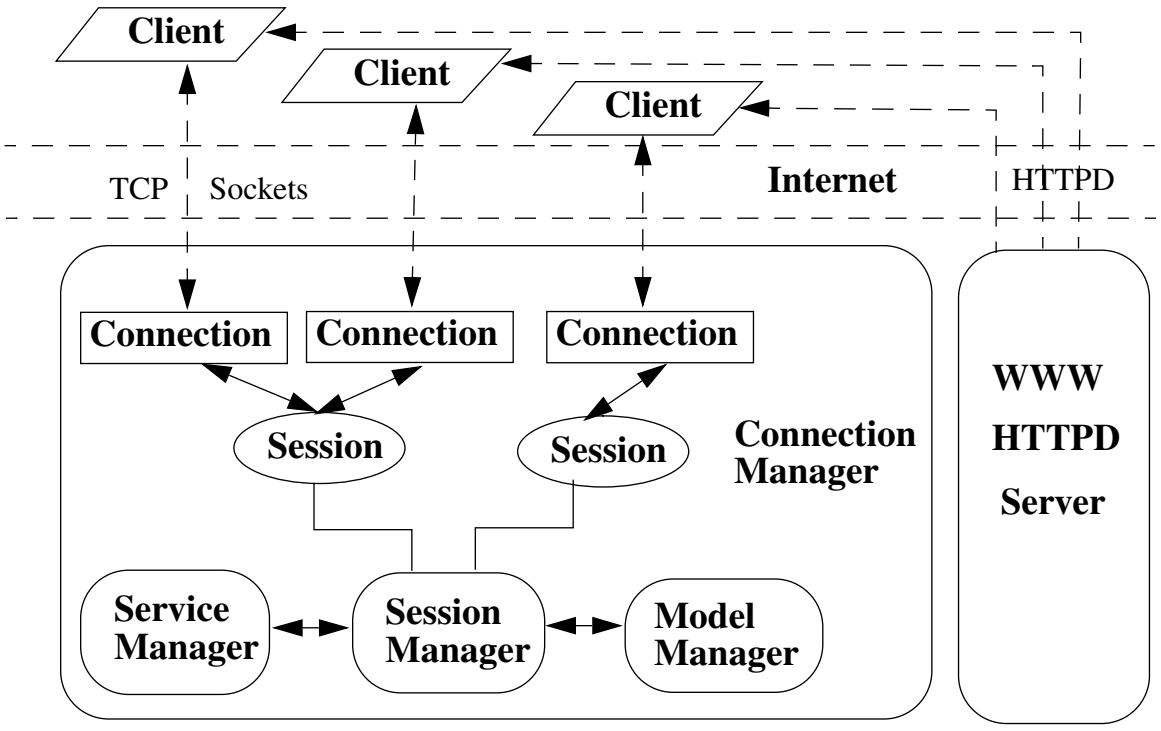
5.1 Collaboration Overview

Collaborative Mocha (CM) provides a What You See Is What I See (WYSIWIS) synchronized collaborative environment. Users of CM share a common working space with other users. In order to support collaboration we found necessary to extend Mocha's architecture and to introduce some changes in the system implementation. These changes and extensions were applied to the server, the clients, the communication protocol and the framework.

5.2 Changes in the Collaborative Server

In Collaborative Mocha the client-server model is extended to support a synchronous collaborative environment. The server role is to arbitrate and coordinate the interaction between the different clients, and to provide services to individual clients.

In order to provide a shared environment it is necessary to support a common context shared by multiple users in a joint session. NC Mocha sessions are strictly single-user, and each session runs on a separate process. The session is simply a single-socket connection between a forked copy of the server and a client.



4.6 Animation Protocol

Mocha's animation protocol provides a common language for data communication between the clients and the server in the environment. This protocol is based on the LEDA library ASCII file format.

By using the LEDA file format directly as the animation protocol we obtained a serviceable description for the geometric objects supported by the system without having to implement it from scratch, thus reducing the development time for the system. The server can execute the geometric algorithms to provide the service requested by the client directly without additional mapping between the geometric object described in the protocol and the data structure used by the algorithm.

On the downside the LEDA file format doesn't provide ways of expressing atomic operations on the geometric objects. In case of a change in a given geometric object, however small the whole object has to be sent over through the communication, producing overhead and reducing the performance of the system. This overhead is not significant in the NC system but it is more important in the CM version because of the additional overhead of keep the different clients sharing a session synchronized.

In CM the animation protocol has been extended to support atomic operations, and to support additional attributes for the geometric objects.

4.6.1 User events

The user events include all the changes introduced in the geometric model by user interaction. Some examples are manipulation of a graph (nodes or edges), editing of points in a pointset, etc.

4.6.2 Geometric objects

Examples of geometric objects supported by the protocol include points, point sets, nodes, edges, graphs, planar graphs, polygons etc. This geometric objects need a common description to support communication between the client and the server, which is provided by the protocol.

4.6.3 Animation Events

The animation events correspond to the "interesting" events of BALSA and TANGO, described as images, locations, transitions and paths. In NC we animated not the algorithm run itself but the final result of the algorithm when applied to the real-time interaction of the user. In Collaborative Mocha we extend the framework to support graph algorithm animations, as will be described in the section 5.4.3.

canvas, where the user can define geometric objects and interact with them, and a tool panel, where the user can select different options of operation and perform some activities.

4.4.4 Animator

The animator maintains the model in response to both user interactions (local and remote, through the collaboration protocol) and animation events)(through the animation protocol).

4.5 Services

The geometry server component of the Mocha system provides several services for solving problems of computational geometry and graph drawing. This services are based upon an algorithm library which includes algorithms provided by the LEDA library and algorithms written by the mocha authors. The algorithms can be animated or not-animated.

In addition to geometry services CM provides session management services. These services include creating new sessions, adding or removing users from a given session, querying the session for a list of users, saving or restoring a session from disk, etc.

.Among the LEDA services provided by Mocha we can mention Convex Hull, Voronoi diagrams and Delaunay triangulation diagrams. Other services provided include a big beta proximity graph service and an animated search algorithms service.

- Convex Hull

The convex hull of a finite set of points S in the plane is defined as the convex polygon whose vertices belong to S and such that any point of S that is not a vertex lies in the interior of the polygon.

- Voronoi diagrams

The voronoi diagram of a finite set of points S is a planar subdivision of the plane into convex regions (voronoi regions) such that each region is associated to one distinct element of S. For each point p in the voronoi region associated to a an element q of S, p is closer to q than to any other element of S.

- Delaunay triangulation

The delaunay triangulation of a finite set of points S is the geometric dual of the voronoi diagram of S, i.e. the graph obtained by connecting two elements of S if their Voronoi regions share an edge.

- Big Beta proximity graphs

The proximity graph exhibits a relation between points by connecting pairs of points that are deemed close by some proximity measure.

The NC protocol supported is the LEDA ASCII file format. The geometric objects in the system are the high level data structures defined in the LEDA library, like points, graphs, edges etc. The annotated algorithms provided by the server are LEDA defined algorithms like Convex Hull, Voronoi diagrams, Delaunay Triangulation diagrams. A Proximity region service is also provided by the geometry server.

Since in the NC model each “session” is single-user and corresponds in effect to a separate process handling the socket connection with a given client this server doesn’t provide shared context between clients. Each process is ignorant am independent of any other processes handling additional clients. The server doesn’t keep state between services call, each call for a service must provide the appropriate input data.

The communication in the NC server is always synchronous, following a request-reply format. In the CM model the communication supported includes synchronous, incremental, asynchronous and active server.

4.4 Client Implementation

4.4.1 Web Browser

Web browsers are quickly becoming a standard application familiar to most computer users. Some of the more popular operating systems like MS Windows 95 are incorporating web browsers as an integral part of the system. As a result many users are familiar with browsers and use them regularly.

Mocha clients are java bytecode applications embedded into WWW pages. In order to access a Mocha client the user must simply “point” his web browser to a given URL. There is no need to installed additional software in order use the system.

This makes the system more accessible to users, and avoids requirements of specific hardware or software platforms. Future advances like the proposed Network Computer could make web based applications even more popular and accessible by reducing high-cost hardware requirements.

4.4.2 MVC paradigm visualization

The model-view-controller separates the model from its representation, allowing multiple views of a common underlaying model, and separates the interaction into the controller layer. In Mocha we extend the conventional use of MVC by partitioning both the model and the controller between the client and the server. The views are fully handled by the client.

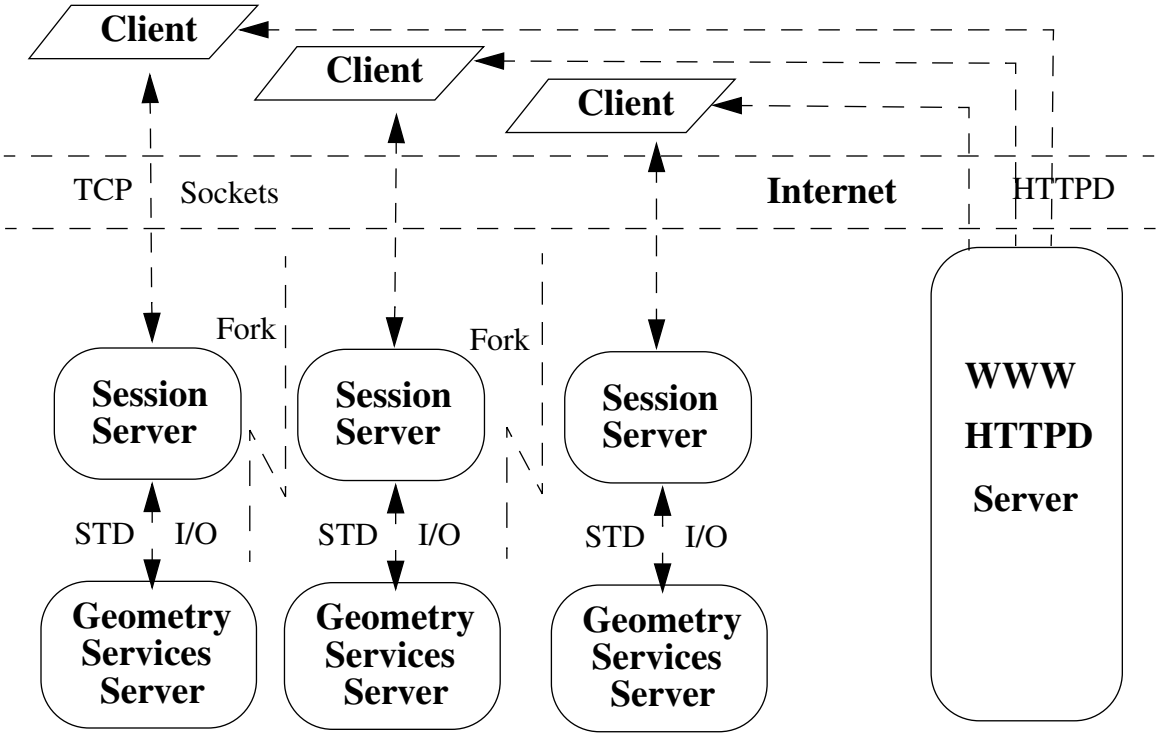
4.4.3 GUI

The GUI supports the view and controller of the MVC paradigm. It is written in terms of java and the java AWT GUI framework. The GUI of the Mocha clients include a geometry

accommodate new clients. In this original implementation there was no support for shared common state between sessions or inter-client communication.

The use of a separate, general-purpose session manager simplified the development of the first prototype of the system. However it imposed certain limitations on the system, namely in the impossibility of sharing state between different clients and the lack of persistence between sessions.

In order to extend the Mocha system to support collaboration this scheme was discarded and a integrated session manager was developed and it was consolidated into the CM geometry server. These changes on the Mocha architecture will be discussed in detail in the next chapter.



NC Geometry Server + Session Server

4.3.2 NC Geometry Server:

The geometry server provides geometry services. It includes a *model manager,* which supports geometric objects, a *protocol manager,* which supports the communication protocol and the *geometric service implementations*, i.e. the annotated algorithms. These components are based on the LEDA library, a C/C++ library that provides high-level data structures and computational geometry algorithms.

that can be embedded in the HTML pages. The author can also take advantage of the hyperlink capabilities of the WWW to add references to related work, etc.

4.2.4 Extensibility

Mocha is designed to provide an extensible framework suited for developing algorithm animation applications and collaborative applications in general. This framework includes classes for computational geometry, communication components, user interface components, and an application specific protocol. This protocol supports geometric objects, user generated events, and animation events.

4.2.5 Collaboration

Collaboration was not one of the original primary goals of the system. However the original design proved well suited for being extended into a collaborative environment. This extensions required some changes in the implementing of the system but not fundamental changes in either the architecture or the design of Mocha.

4.2.6 Advanced Interactions

Mocha provides advanced interactions between the user and the geometric model and algorithms that the system provides, and also between the users collaborating on a shared working space. Geometric data structures and algorithms have many parameters and attributes, and require robust interaction. In addition, collaborative environment required ways of organizing the joint multiple-user interaction with shared data to achieve common goals.

4.3 NC Server Implementation

The server implementation in the NC prototype of the system uses 2 separate and external components: a general-purpose session manager and a geometry services server.

4.3.1 External Session Manager:

The session manager supports the creation of state through the use of processes and sockets. The original implementation of the session manager used a general-purpose socket server that provides access to TCP sockets from the unix shell.This server binds a server socket to the given port on the local host and accepts a connection. When a client connects to this socket, all data read from the socket is written to stdout, data read from stdin is written to the socket.

The input data read from the socket is written to stdout and the geometry services server is called. The geometry server processes the data, computes the geometry output requested and writes the result to stdin where it is read from the socket server and written back to the socket.Each clients uses a different process. New processes are spawned as needed to

each client connects to a different copy of the server (using sockets) running on its own process. There is no communication between peers.

In CM every client establishes a TCP socket connection to a Central Server. Communication between "peers" goes first to the Server, which in turn broadcasts the message to the appropriate clients. We use an application-specific protocol to handle the communication.

4.1.3 Heterogeneity

The client side is implemented in Java. We are able to exploit and benefit from Java portability and cross platform availability. The system can be accessed from any Java-enabled web browser. Specific hardware requirements are eliminated, and the system is accessible from most platforms. This also means that users using CM on a specific platform will be able to communicate and collaborate with other users on a different platform transparently and effortlessly.

4.1.4 Framework

Frameworks provide a cooperating classes providing reusability of design and implementation over the given domain. Mocha supplies a extensible framework for building algorithm animation and collaboration tools.

The Mocha framework is built on top of the Java framework. It provides high-level objects that encapsulate some of the complexity of dealing directly with the Java library components, like the java.awt GUI library or the java.net communication library.

4.2 Design Goals

4.2.1 Security

Java provides support for security on the user side. On the provider side, security is guaranteed both by Java and the design of the algorithm servers. The code is also protected since it is distributed only in bytecode form, and the source remains undistributed.

4.2.2 Accessibility

We want Mocha to be as accessible as possible, by achieving cross-platform portability and hardware independence. This approach eliminates specific software and hardware requirements and most common porting or installation difficulties that could reduce the number of potential users of Mocha.

4.2.3 Authoring

Mocha provides full support of the World Wide Web and its standards. Authors and users of algorithm animations can place the desired applet as another component of a HTML document. The applets can be combined with images, sound and other multimedia types

4.0 Mocha Architecture

4.1 General overview

Mocha[Tamassia et al] implements a distributed architecture for algorithm animation. Mocha partitions the algorithm animation environment by executing the algorithm in the provider's machine while the UI and animation is executed in the user's machine. This component partitioning maximizes accessibility, ease of authoring, and code protection.

Mocha leverages existing tools, standards and architectures in order to take advantage of current open solutions and. avoid the waste of effort involved in duplicating available technology by implementing proprietary solutions.

Thus the Mocha model employs a configuration of software frameworks, client-server portioning and layered protocols to provide openness and accessibility.

Mocha was originally conceived as a single-user system. In this work we extend Mocha to implement a collaborative environment for education. We will refer to this extended version as Collaborative Mocha (CM for short) in order to establish the differences between the original system and the multi-user shared environment. The original non-collaborative system will be referred to as NC Mocha. The system as a whole (both versions) will be simply called Mocha.

4.1.1 Client-server model

The Mocha system is based on a client-server paradigm. The WWW server distributes Java code and multimedia objects to the GUI on the users' machine by means of a HTTP protocol. In addition a application server provides miscellaneous services including geometry, algorithm, and session services.

The application server executes the algorithms and any computations necessary. The client handles the GUI of the system, interacts with the user and displays the results of the algorithms by means of an animator. In CM the server also keeps synchronized all the clients sharing a common session and arbitrates the joint user interaction with the geometric model.

The communication between the application server and the clients is done by means of an application specific protocol. The protocol supports user events, geometric objects and animation events. In Collaborative Mocha this animation protocol is extended with a session protocol that supports session management.

4.1.2 Distribution mechanism

Communication between Mocha components is achieved by using the standard WWW's HTTP protocol and by using an application-specific protocol, in a distributed client-server scheme. Mocha uses TCP sockets as the primary distribution mechanism. In NC Mocha

visualization of multi-dimensional geometric data. The dynamic component of Shastra is a runtime environment that provides runtime support for the conferenced tools.

The CSCW infrastructure of Shastra facilitates the creation of collaborative multimedia tools. Shastra is based on an abstract tool architecture that enables inter-tool communication and cooperation. It supports remote task invocation and brokering. Tools are considered objects that provide specific functionality and exchange messages (automatically or under user command) to request other objects to perform operations.

Shastra provides both tool-level and user-level cooperation. Tool-level cooperation is achieved by specifying architectural guidelines and providing communication facilities for inter-tool cooperation and information exchange. User-level cooperation is achieved by providing the facilities necessary to build collaborative tools where users are empowered to collaborate and cooperate in order to achieve a common goal.

Shastra tools are abstractly composed of three layers. The core layer contains the application-specific tool functionality. The interfaces layer includes a GUI, network and ASCII interface. The GUI interface allows the user to access the tool functionality by using a graphical front-end. The ASCII interface is a shell-like front end for the tool. The network interface allows Shastra tools in a common environment to communicate with each other.

The Shastra run-time environment is composed of several components. Shastra tools are the building blocks for creating the run-system. Kernels and Session managers are managing tools, responsible for maintaining the distributed and collaborative environment. Brokers offer distribution and task brokering facilities. Toolkits provide scientific design and manipulation functionality. Service tools provide mechanisms for communication and animation.

Several Shastra applications have been developed for different purposes, including Graphics, Multimedia, Games, Visualization, Surface Design, Solid Sculpting and Design Prototyping. Collaborative Graphics applications include Sha-Draw, a 2D sketching tool, and Sha-Poly, a 3D visualization tool. The Multimedia applications include Sha-Talk a text “chat” tool, Sha-Phone, an audio-conferencing tool and Sha-Video, a video-conferencing tool. The Sha-Vaidak tool is used for 3D Visualization.

Other Shastra applications are a combination of different tools. The Surface Design environment uses 3 tools: a Kernel, and the Shilp and Ganith toolkits. This is an example of multi-user collaborative aware task sharing in the context of Shastra. This permits a group of users to collectively smooth out a rough polyhedral model of a car, for example by fitting continuous smooth patches of using hermite interpolation. The Ganith toolkit provides algebraic manipulations. The Shilp toolkit provides boundary representation-based solid modeling.

The Solid Sculpting environment uses 2 shastra tools, the Shilp and Sculpt toolkits. Sculpt is optimized to provide a set of operations (union, intersection, difference and complement) over polyhedral geometric models. Together they provide a powerful multi-user interactive solid modeling facility based on Constructive Solid Geometry (CSG).

large co-present groups collaborative environments can be based on a local-area network (LAN) or an intranet. Some sample applications are described in [Brikcer et al].

Computer Supported Collaborative Work (CSCW) applications are similar to CSCL, only the term is more general and refers to work-related activities including business, academia, research, government work etc.

CSCW applications can be divided in synchronous and asynchronous. Among the asynchronous systems we can mention messaging systems and electronic mail (email). Email is the oldest and most proven of the electronic collaboration vehicles. Originally limited to text, email now supports multimedia objects thanks to new standards like Intenet MIME. Furthermore the proposed Computational Mail would allow people to communicate not only through the exchange of passive data, but by executable programs.

Lotus Notes, probably one of the most successful commercial CSCW systems available is also an example of asynchronous system. It provides protected information sharing but has limited support for collaboration, with no systematic way of providing user awareness, limited support for conflict management, and asynchronous update propagation.

Some examples of synchronous collaboration environments for the Internet and the WWW include WebDesk[Parnes et al], a distributed conference environment, Yarn[Woo, Rees], an electronic meeting system, Mushroom[Kindberg], a framework for collaboration and interaction across the Internet. Other examples mentioned previously include ARGO, DIVE, etc.

A prime example of a research CSCW is Shastra from Purdue University. Shastra is described in detail below.

Collaborative Mocha, the environment and framework presented in this paper also falls in this category of collaborative system.

3.6.1 Purdue University Shastra Project

Purdue University's SHASTRA [Bajaj 1995] Project is directed at research in CSCW based geometric modeling, simulation, interrogative visualization and design prototyping environments. SHASTRA goal is to build multi-user collaborative scientific design and analysis environments. The name comes from the Sanskrit word for Science.

Shastra is an extensible, distributed and collaborative geometric design and scientific manipulation environment. It consists of a static and a dynamic component. The static component is a CSCW infrastructure for building CSCW tools, called the Shastra layer. It is composed of several substrates. The connection and distribution substrate facilitates inter-tool cooperation. The communication substrate supports data sharing and transport of multimedia information. The collaboration substrate supports synchronous multi-user tools by providing session management and interaction control.

Shastra also provides a numeric, symbolic, graphics and multimedia substrate. It enables rapid prototyping and developing of collaborative tools for creation, manipulation and

sions to support URLs and WWW linking between VRML and HTML documents. Open Inventor is a OO 3D graphics toolkit developed by SGI. Although the name can be misleading (it is not really a programming language, and “virtual reality” is a relative term) VRML provides a common framework for describing 3D worlds in the WWW, and allowing some standard user interaction with these worlds.

VRML 1.0 was designed to be a minimal starting point for a much larger vision. As such it has many limitations. It only provides means for describing static 3D scenes with no support for behavior, complex user interactions, collision detection, animation, etc. This limitations reduce considerably the potential use of VRML for advanced applications.

At this time a draft for the VRML 2.0 specification is being prepared. The proposed VRML 2.0 specs will include extensions to VRML 1.0 including support for interaction, animation and prototyping, as well as enhancements for static worlds, such as sound, additional nodes, etc.

VRML has the advantage that it is supported by the existing web browsers. VRML can be supported directly by web browsers, or by extensions to the browser using technology like plug-ins or “helper” applications. VRML also coexists with all the existing WWW and Internet infrastructure, like HTML pages and CGI-bin scripts, and with the existing protocols like httpd.

The biggest drawback of VRML is the high-bandwidth that it requires. VRML performance with low band connections like 28.8 modems has been less than desirable, even with language constructs like LOD (level of detail) designed to optimize performance.

3.6 CSCL and CSCW

Computer Supported Collaborative Learning (CSCL) is designed to “encourage cooperation, discussion of ideas, resolution of cognitive conflicts, and promote problem solving and higher-order thinking skills”. [Brikcer et al]

CSCL systems can be classified in two types: those supporting cooperation at a distance and those supporting a co-present (face to face) environment. Low-end cooperation at a distance systems are often restricted by low degrees of apparent presence provided by the system and lack of actual contact with other students. More sophisticated system relaying on video teleconferencing reduce this problem but are considerably more expensive.

Some examples of CSCL at a distance systems in the Internet include some systems mentioned in previous sections, like the Global Network Academy [GNN FAQ], and the NSF-STC Center for Scientific Visualization and Computer Graphics Video-Conferencing network. Another example is the Learning Circles environment [Riel 1990] developed by the AT&T Learning Network.

Co-present collaboration allows students to interact directly and offers more effective communication. Participants can see each other’s faces, expressions and gestures. For

tions let collaborators interact jointly with a common document, however they are typically forced to fall back on text or video for communication between users.

Virtual Collaboration uses virtual reality environments to support collaboration. Virtual reality (VR) systems are defined as “support systems for interaction and navigation in a 3D workspace” [SICS]. Users perceive the environment as a “world”, he is given a location and a presence in the 3D environment, and he can interact with objects and other users present as well. Virtual reality has enormous potential for providing natural and transparent means for manipulating objects, accessing information and collaboration between remote users.

Among the current research in VR based collaborative environments we can mention DIVE [Benford, Fahlen][SICS], a distributed interactive virtual environment system developed by SICS. This system provides a high-capacity distribution infrastructure, a framework for developing and supporting collaborative VR applications, and a user, object and interaction scalable model.

Another project on VR collaboration is a multi-user VE (Virtual Environment) system [Conner et al] developed by SRI. This system uses a hand-held display (HHD) to provide an interface into the shared VE. The HHD is carried by participants and displays a view of the virtual world as seen through a small window. The HHD is a LCD display whose position and orientation are captured by a six-degrees of freedom (6D) tracker.

There is also interest in the NSF-STC Center for Computer Graphics and Scientific Visualization channeled through a telecollaboration project that aims to “make telecollaboration as real as being there” [Conner et al] by using distributed VR environments for collaboration.

3.5.1 Inmersive Environments

In inmersive VR environments the user is isolated from the real world by means of inmersive devices like VR helmets and BOOM devices. These devices capture the user’s head location and orientation in space, and adjust the image displayed accordingly. Another inmersive experience is provided by CAVE-like systems, which are cubicles with display screen faces surrounding the viewer.

These setups provide a very realistic VR experience, but they are usually very costly, difficult to carry around, and intrusive.

Other alternatives like fixed displays like desktop monitors or projector-like displays are considerably less expensive but have the drawback that they break the direct relation between the user’s position in the virtual space and its perception through the display.

3.5.2 VRML

VRML stands for “Virtual Reality Modeling Language” [VRML FAQ]. The original VRML 1.0 specification is a subset of the Open Inventor ASCII file format with exten-

The MBONE is an effort to construct an infrastructure (physically based on the internet) to support audio and video conferencing. It uses a network of routers (mrouters) that support multicasting. These mrouters are either upgraded commercial routers or dedicated workstation typically running with customized kernels in parallel with more standard routers.

MBONE is in an experimental stage, and is nowhere near product-level robustness. However there is interest on the part of network vendors to create commercial products based on this technology.

3.4.2 CU SEE-Me

CU SEE-Me[CU-SEEME FAQ] is a system developed at Cornell University that allows low bandwidth, low cost video-conferencing. CU-SEE Me supports either one-to-one video conferencing or, by using a "reflector" software, one-to-many, several-to-several, or several-to-many video conferencing. Each participant can choose to be a sender, a receiver, or both.

CU SEE-Me achieves its low bandwidth requirements by restricting the video image size to low resolution (either 320x240 or 160x120) and low color depth (4 bit grayscale). It can be used even with low speed modem connections (14.4 or 28.8bps), and requires low cost cameras. The client software can run on MS Windows based PCs or Apple Macintosh PCs. The reflector software requires Unix based workstations.

An example of CU SEE-Me used as an educational tool is the experience of the course English 309M: Computers & Writing [Warshauer], taught at the University of Texas at Austin. In this course, some classes are devoted to experiencing a MUD session and a CU See-Me session. These events are described as follows: "the audio/video/text based CU-SeeMe session becomes a group, cooperatively authored and performed, improv production. Likewise, when viewed as an event, the text based MUDtranscript which is left after the time of the original interaction becomes a group collaboratively authored and performed improv playscript."

CU SEE-Me allows the students in this session to experience theater in a new way, crossing the border between audience and performer and redefining each of them.

In conclusion this technology has the benefit of low requirements in cost and bandwidth. However it suffers from some limitations regarding the quality of the video and audio obtained.

3.5 Virtual (3D) Collaboration

Traditional collaboration systems, based on shared applications or audio/video-conferencing have limitations on the degree of collaboration supported. The standard telephone provides good audio with some sense of presence, but it doesn't support visual information. Televideo setups lets users see each other, but TV images are low resolution and doesn't let speakers move around each other or around an object or discussion. Shared applica-

graphic location. Ideally audio conferencing should provide uninterrupted sound. This means that users should not have to repeat themselves because of dropout.

Sound should be transmitted with low latency, so users don't interrupt one another or have to modify their natural behavior to avoid such interruptions.

Video conferencing extends audio conferencing one step further to provide real-time video-based face to face remote meetings. As an example environment that makes extensive use of videoconferencing for collaboration we can mention the National Science Foundation Science and Technology Center for Computer Graphics and Scientific Visualization, a consortium of research groups from 5 universities (Brown U., UNC Chapel Hill, Caltech, Cornell and U. of Utah).

A dedicated high-bandwidth (T-1 capacity) communications infrastructure between the member sites of the Center furnishes simultaneous interactive video and data exchange on demand, providing an effective distributed collaboration environment for research and education. The sites schedule regular video-conference meetings to coordinate research between the different groups, and weekly lectures are broadcast from a given site, covering the ongoing research on that particular group.

The high bandwidth of the system allows high quality full-rate real time video. This is a good example of the state of the art in video-conference technology but the high cost would make it prohibitive for most organizations.

A inexpensive alternative is offered by video/audio conferencing software components as the ones bundled with the most recent releases of Netscape and Microsoft Web Browsers, namely MS Netmeeting and Netscape CoolTalk (see section 3.3.3). The lower quality is matched by extremely low costs.

Some CSCW/CSCL environments (see section 3.6) offer their own video/audio conferencing tools integrated with the environment. An example is Sha-video and Sha-talk, part of the Shastra Collaboration environment of Purdue University [Bajaj 1995]. See section 3.6.1.

3.4.1 MBONE

"The MBONE is an outgrowth of the first two IETF "audiocast" experiments in which live audio and video were multicast from the IETF meeting site to destinations around the world. The idea is to construct a semi-permanent IP multicast testbed to carry the IETF transmissions and support continued experimentation between meetings. This is a cooperative, volunteer effort. "[MBONE FAQ]

The IETF is the Internet Engineer Task Force, the protocol engineering and development arm of the Internet. The IETF is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet.

3.3.3 Groupware

Applications developed specifically for multiple users (so called “groupware”) offer additional flexibility, transparency and greater functionality as a collaborative environment. However most applications are single-user, and don’t contemplate collaboration at all.

Groupware has been defined in [Gustafson] as “a class of computer technologies that enables information sharing, exchange, coordination and collaboration between two or more people. Groupware is marked by three characteristics: it is designed to be used by a group of people, it has little or no value when used alone, and it removes barriers of time and space.”

Groupware applications can gain extra-leverage from the knowledge that they are being shared. Single user applications shared by means like X-windows server proxies only provide very limited and generic forms of interaction between remote users. By contrast groupware applications designed from the onset to support multiple users can offer complex and rich forms of joint interaction that can be tailored specifically to the joint activity that the application supports.

Groupware is quickly becoming a rapidly growing market. Among some of the most successful groupware commercial products we can mention Lotus Notes and Microsoft Exchange. Netscape is also actively moving in the groupware direction with its Server platform grouped in its product line SuiteSpot, which includes Mail, News, Calendar, Directory, Media, and Web servers among others.

Traditional text-based groupware tools are giving way to new Multimedia-based groupware applications, which support audio and/or video conferencing and shared whiteboard in addition to document-sharing.

An example of this new breed of Multimedia Groupware is the new Netscape “Communicator” (v4.0) release of their web browser (currently at the beta stage) includes new Collabra text-collaboration support, audio-conferencing and multi-user shared whiteboard (previously known as Netscape Cooltalk) HTML-based email, and Usenet newsgroup support. Collabra, email and newsgroup support is integrated into a new Messages Center.

Microsoft is also moving its Web Browser, Internet Explorer in the direction of Groupware. The full installation of Explorer bundles NetMeeting, an audio and video conferencing tool with support for shared whiteboard, file sharing and planned application-sharing support. Explorer also includes Usenet newsgroup support and text-based email capabilities. NetMeeting conferencing is based on Internet standards H.323, (H263 for video) and T120 for whiteboard and shared applications.

3.4 Video and Audio Conferencing

One important aspect of collaboration is the ability to communicate ideas clearly and easily. Speech is the natural mechanism that we use for communication. Audio conferencing provides an audio environment where remote users can speak be heard regardless of geo-

3.3 Shared Applications

Shared applications are a fundamental component of remote collaboration. Fax, telephones and conventional telecommunications are useful tools that have helped autonomous workers to communicate with remote peers. However they provide only limited means of communication that don't contemplate more complex forms of interactions required for activities like joint design or strategic meetings. Audio and video conferencing can meet some of these needs, specifically remote meetings. However activities like design require special tools or applications, and a collaborative environment that supports these activities will require shared versions of these applications as well.

There are three main different approaches to sharing applications: windows system-based replication, toolkit-based replication, and groupware.

3.3.1 Window system-based replication

Window system-based replication attempts to bridge single-user programs into collaborative environments. These systems implement a "shared" X-server that multiplexes the windows of regular, unmodified X applications onto multiple workstation displays.

Among these systems we can mention XMX, developed by John Bazik of Brown University, XTV, developed by Kenny Jeffay of UNC Chapel Hill and Hussein Abdel-Wahab of Old Dominion University. They allow multiple users to communicate and use jointly a shared application in a structured fashion. Typically only one user at a time has control over the application, described as "having the floor". If another user wants the floor it has to be yielded by its current owner. One particular user is usually the coordinator (sometimes called "chairman" of the collaborative session) and has special privileges.

This approach has the advantage that will allow any single-user X-windows based application to be used in a collaborative fashion. However it has several disadvantages as well. The interaction possible with these systems is very limited: only one user can have control at a time. It is also platform-dependent, since it is limited to X-based Unix applications.

3.3.2 Toolkit-based replication

A toolkit it defined in [Gamma et al] as "a set of related and reusable classes designed to provide useful, general-purpose functionality.". Toolkits emphasize code reuse, let programmers avoid recoding common functionality, and simplify the development of applications. An example of a windows-system independent, object -oriented toolkit that supports collaboration is Trestle, developed by DEC and used in their ARGO[Gajewska et al] collaborative environment.

Toolkit-based replication has the advantage that simplifies the development of multi-user applications, by providing common mechanisms for collaboration. These applications can support both a single-user and multi-user environments, so they don't have to be strictly groupware. However the toolkit provides a common framework for developing applications with different levels of collaboration support, from group-aware to groupware.

The MOO object oriented programming language embedded into the MOO system makes the environment easily extensible Any user can potentially extend existing objects or create new ones. This flexibility, coupled with MOOs low bandwidth requirements has contributed to the popularity of these systems.

MOOs are being used extensively in CSCL and CSCW. Among ongoing projects in CSCL based on MOOs we can mention the Globewide Network Academy [GNN FAQ], who intends to establish a competitive market for distance learning courses and programs. Their "interactive classroom" is a prime example of a MOO used for education.

Hypermedia and writing are also areas in which MOOs have proved useful tools. A system developed in Brown University uses MOOs to create a collaborative hypermedia environment to support groups of writers in writing and publishing hypermedia fiction. The system uses the WWW as a publishing platform and to provide authoring tools based on HTML forms.

In addition MOOs are breaking free of their earlier limitations on media (as text-only systems). New systems like Jupiter [Curtis et al] developed by Xerox PARC extend MOOs to incorporate support for multimedia. Jupiter enhances the MOO environment by providing user-definable windows and support for real-time audio and video streams. Jupiter at present only supports X-windows and Unix-based applications. This system is being used to develop collaborative environments like Astro VR [Benford et al], a multi-user environment for research in astrophysics developed as a joint project between the University of Seattle, Xerox PARC and the Infrared Processing and Analysis Center.

In conclusion, MOOs are flexible and extensible systems. In their simplest incarnation they also require low-bandwidth and are relatively cross-platform (specially with WWW-based clients), but they are limited to text and do not support multimedia. The more recent extensions to MOOs add multimedia support, but require higher bandwidth and lose to some extent the platform independence of more basic MOOs.

3.2 Collaborative Creation

Under this category we find some collaborative systems for artistic creation. Each participant contributes to a particular aspect of the overall creation. The creation itself can be a visual, literary, audio or video work, among other types.

Some examples of visual collaborative creation systems are the Hyp-Art Project, the Collaborative Visual Art 5x5 Project, the Otis:Synergy GRID Project, etc. The common theme in these systems is dividing a image in segments each of which will be the creation of a different participant.

Other examples of collaborative creation are hyper-text based system for collaborative writing, like Hypertext Hotel MOO and WAXweb MOO. Another technique is to create "genetic" art (visual, audio, etc.) by letting the participants choose "mutations" of the artistic creation. Some examples are Interactive Genetic Art I & II, The Morph Lab, etc.

3.0 Collaboration in the Internet

The Internet presents a fertile ground for the development of collaborative environments. There is substantial interest both in industry and in academia to explore the potential of collaboration. Traditional “groupware” has focused on asynchronous forms of text-based collaboration, like email. Past and present technical limitations like low bandwidth imposed certain limits on the types of collaboration that were practical to implement. Text based applications typically require less bandwidth than graphical programs. Moreover GUI based applications are relatively recent developments.

The infrastructure of the internet has its origin in military networks like ARPANET that preceded by several decades the development of GUIs. Early systems for organizing and sharing information in the internet, like Gopher used text-based user interfaces. However GUIs, specially WIMP-based user interfaces (Windows, Icons, Mouse, Pointers) have become prevalent in recent times, and the internet has not escaped its influence. Thus the World Wide Web now offers a more user friendly, graphical front-end to the internet that has been a considerable factor on the current popularity of the internet.

New advances in communication technology have made possible to explore new forms of collaboration, supporting different levels of synchronization. We will discuss next several of the current efforts in collaborative environments on the internet, and classify them according to their characteristics. We will then examine how Mocha fits into the overall collaboration work currently ongoing in the Internet.

3.1 MOOs and MUDs

MOOs (MUD, Object Oriented) are defined in [Reinhardt] as “a network-accessible, multi-user, programmable, interactive system well-suited to the construction of text-based adventure games, conferencing systems, and other collaborative software. “. MUDs [Curtis] (Multi User Dungeons) originally were conceived as a vehicle for conducting multiple user role playing games. They provided a simple text-based interface that could be accessed by standard internet telnet applications or by MUD specific clients.

Users assume the persona of an “avatar”, or character. They can define the identity of their characters and control their actions in a “virtual” world composed of rooms and items described by textual descriptions. The user issues commands using a command-line interface. These commands may cause a change in the “reality” of the MUD or simply query and report on the current state of the reality.

MOOs are an object oriented extension to the original MUD concept. MOOs have 2 main components: the server, and the database. The server, which is written in a standard programming language, manages network connections, provides access to the database, and executes commands, other programs written in the MOO programming language. The database contains descriptions of all the objects in the MOO “reality”. The behavior of objects in the database is defined by verbs (methods) and properties (data). Objects, verbs and properties can be created and altered from inside the environment.

ally any collaborative environment based on the internet should leverage these different technologies in order to minimize developments efforts and take advantage of the existing infrastructure.

Algorithm animation has proved to be a powerful pedagogical tool in Computer Science. It can be used in both in a individual basis or in groups (for example in class demonstrations), which make it suitable for collaborative learning. Algorithm animation provides a visual representation of data structures and algorithm execution that can simplify algorithm learning and understanding. It can be used also in other task like software visualization and debugging.

In order to provide a CSCL environment for algorithm animation we extend Mocha [Tamassia et al], a model for algorithm animation over the World Wide Web. Mocha is a distributed model with a client-server architecture that optimally partitions the typical components of an algorithm animation system. On the Mocha model, only the user interaction and the display is handled on the user machine, while the algorithm itself will be run on a server running on the provider's machine.

Mocha synchronous collaborative environment provides WYSIWIS (what you see is what I see) full synchronization. The server arbitrates and coordinates the interaction between the different clients, and provides services to individual clients. The collaborative interaction between users includes a shared whiteboard, and a textual conversation or "chat" panel.

Mocha provides high levels of security, protects the algorithm code, places a light communication load on the internet, and allows user with limited computing resources to access animations of computationally expensive algorithms. The user interface combines responsiveness and user friendliness with the multimedia authoring capabilities of HTML.

Computational Geometry is one of the topics in Computer Science which can benefit from the didactic use of algorithm animation. We have implemented a prototype of a shared geometry whiteboard using the Mocha framework. This prototype provides an intuitive, more accessible understanding of computational geometry concepts like convex hull diagrams, voronoi regions, delaunay triangulations and big beta proximity graphs, among others.

By providing a collaborative environment for computational geometry we make possible the establishment of learning at distance schemes based on a virtual classroom that takes advantages of the existing infrastructure of the WWW.

2.0 Introduction and Overview

In the last decade a considerable amount of work has been dedicated to study the potential of CSCL. However it is widely held that CSCL has yet to reach fully this potential. New developments in technology promise to change this situation and make CSCL more successful and prevalent than it has been in the past, as has been reported in [Meyer, Blair, Hader].

These developments fall in three broad categories: networking, multimedia, and mobility. Networking refers to LANs, WANs, and on-line services. The internet in particular has seen explosive growth and currently enjoys a very high profile. The rising popularity and availability of network providers, combined with advances in communications have made networking more accessible and faster than ever before.

Multimedia has also become cheaper and more widely available. New media has made possible new forms of visualization and interaction, and new possibilities for education and learning. Mobility has been provided as a consequence of both networking (including telephony advances like cellular networks) and the miniaturization trend that has produced notebook computers and PDA devices among others. It eliminates some of the traditional physical restrictions on location and infrastructure requirements.

The World Wide Web is a significant factor in the rise in popularity of the internet. It has provided a powerful yet “friendly” graphical interface for the internet that made it more accessible to “soft-core” users. Its hyperlink capabilities make it easy to organize information in multiple multimedia formats, and it has proved to be a highly effective mechanism for sharing distributed information. However, until recently the World Wide Web allowed only restricted forms of interaction, mainly by text-based HTML forms and cgi-bin scripts [Gleeson et al]. These limitations could restrict its appeal as a medium for collaborative learning, although text based systems like internet MOOs and MUDs have been successfully used for a variety of educational purposes, from math to writing teaching. [Benford et al] [Curtis et al]

New advances like the Java language have opened a window of opportunity for advanced interactive applications embedded into multimedia-rich HTML documents. Among Java’s more attractive features we can list its platform-independence and complete multi-platform availability and portability. Java is also object-oriented, and therefore easily extensible, and provides a powerful framework which includes networking, GUI, and multimedia capabilities.

Thus the World Wide Web is being extended from its original unidirectional, asynchronous client-server model into bidirectional synchronized collaborative environments which could prove excellent settings for the development of CSCL.

It is important to note that there are currently several comprehensive, complementary efforts to develop new technologies for the World Wide Web and the Internet. Thus we have in 3D graphics the VRML development; in the interactive front we have Java, JavaScript, and other scripting languages, in multimedia there is ShockWave, etc. Therefore ide-

A collaborative environment for algorithm animation on the WWW

Luis David Lejter

1.0 Abstract

On this paper we extend Mocha, an algorithm animation system for the World Wide Web to incorporate a Computer Supported Collaborative Learning environment. Mocha uses a distributed model based on a client-server architecture that leverages existing network, multimedia and algorithm animation technologies.

Collaborative Mocha provides an extensible framework suitable for developing collaborative pedagogical tools based on algorithm animation. By using existing technologies Mocha takes advantage of current developments in internet technologies and simplifies the development of new tools.

Our CSCL for Mocha features a synchronous collaborative environment, providing a WYSIWIS (What you see is what I see) experience. We cover different aspects of the collaboration environment like awareness, embodiment, visualization, persistence and history, etc.

We describe a shared geometric whiteboard prototype suitable for learning concepts in computational geometry. This prototype can be accessed from any java enabled web browser.

Keywords: algorithm animation, computer supported collaborative learning (CSCL), World Wide Web (WWW), computer supported collaborative work (CSCW).

A collaborative environment for algorithm animation on the WWW

Luis David Lejter

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-97-07
May 1997