Alternative Interfaces for Chat

David Vronay, Marc Smith, and Steven Drucker
Virtual Worlds Group, Microsoft Research
One Microsoft Way, Redmond, WA, 98052
{davevr, masmith, sdrucker} @microsoft.com

ABSTRACT

We describe some common problems experienced by users of computer-based text chat, and show how many of these problems relate to the loss of timing-specific information. We suggest that thinking of chat as a real-time streaming media data type, with status and channel indicators, might solve some of these problems. We then present a number of alternative chat interfaces along with results from user studies comparing and contrasting them both with each other and with the standard chat interface. These studies show some potential, but indicate that more work needs to be done.

KEYWORDS

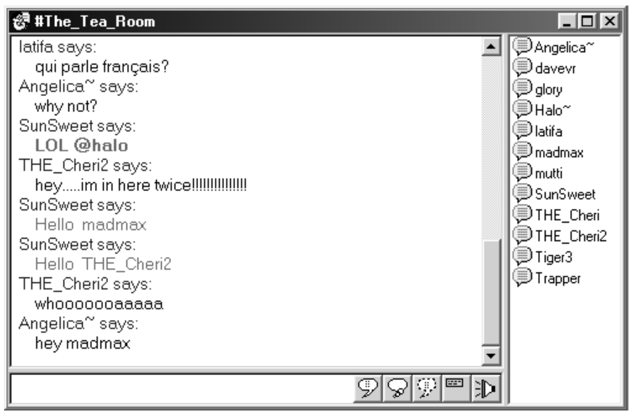
Chat, computer-mediated communication, streaming media, visualization, time-based media

INTRODUCTION

Since the earliest days of computing, people have used computer systems to enable communication. Simple command-line programs such as `write` (and later `finger`) have existed on the earliest time-sharing systems, while computer bulletin board systems (BBS) were flourishing shortly after the introduction of personal computers in the mid-1970s.

The advent of large multi-user systems (such as AOL, CompuServe) and the Internet itself enabled a further style of communication known generally as "Chat". Chat consists of two to twenty or more people who appear together on a common channel of communication (often known as a "channel", a "chat room", or simply a "room"). Any of the participants can type a line of text (a "turn"). By pressing RETURN, the turn can be sent to all of the other participants. Chat has been incredibly popular since its inception, and continues to grow with the expansion of the Internet. Currently there are hundreds of chat servers, each with dozens of active rooms.

Despite (or perhaps due to) its popularity, the user interface presented for chat has remained largely unchanged since its first inception. A typical chat interface consists of a list of participants, a text history window, and a single line for text entry (figure 1).


#The_Tea_Room

latifa says:
qui parle français?
Angelica~ says:
why not?
SunSweet says:
LOL @halo
THE_Cheri2 says:
hey.....im in here twice!!!!!!!!!!!!!!!
SunSweet says:
Hello madmax
SunSweet says:
Hello THE_Cheri2
THE_Cheri2 says:
whoooooooaaaaa
Angelica~ says:
hey madmax

Angelica~
davevr
glory
Halo~
latifa
madmax
mutti
SunSweet
THE_Cheri
THE_Cheri2
Tiger3
Trapper

Figure 1:MS-Chat, a standard chat client

As a communication medium, chat is not without its problems. The conversations in chat tend to be hard to follow. People have trouble identifying speakers and remembering who said what. Within a particular room or channel, there are often multiple conversations or "threads" going on at the same time, and a single user often participates in multiple threads at the same time. As a result, chat conversations tend to be confusing and disjointed, with a lot of repetition and corrections. Studies have suggested that in a typical chat conversation, as many as 40% of the turns are repetitions or corrections to misunderstandings of previous turns [9].

PREVIOUS WORK

There has been a great deal of work on computer-mediated communications (CMC) in the past twenty years. CMC generally considers chat along side of other computer-based communications, such as E-mail. Despite the number of years of research, there is almost no literature evaluating the fundamental user interface of social chat software, and very few examples with actual quantitative data on chat conversations [9].

Much work has been focused on the sociological or linguistic properties of CMC [2][10][12][13][14]. This research typically deals with the ways in which users attempt to overcome the limitations of chat software, and the degree to which these attempts are successful.

CMC is also studied as a component of Computer Supported Collaborative Work (CSCW) [1] [3] [5] [6] [8] [15]. In this context, we see more work on the effect of specific user interface notions or affordances that support groups of people working together to accomplish some task. While some of these notions might be applicable to the more unstructured communication of a social chat, they mostly involve notions of floor control or turn-taking that are not appropriate in the free-for-all of a chat room.

CMC is also frequently discussed in the educational domain [7][9][11]. The interest here seems to be largely based on the increasing number of classes that are being held "on-line", in which CMC serves as the only communication medium available. The bulk of this writing is geared around the particular problems that both instructors and students encounter due to the limitations of CMC.

FACTORS INFLUENCING CHAT EFFICIENCY

There are many factors that impact the efficiency of chat. Here we list several of them, and show how they interrelate to each other.

Lack of recognition – users find it difficult to associate a nickname with what a user has said in the past if they do not already know the speaker. Some clients allow users to set the color and/or style of their text to help out with this problem.(mIRC). Other clients allow the user to choose a picture or cartoon to represent them in a virtual space (MS-Chat, vChat).

Lack of intention indicators – it is difficult or impossible to tell if a particular comment was intended to be directed towards a particular user or turn. In transcript 1, Red's answer is ambiguous and could apply to either Black or Pink's question.

```
Black:  What is your favorite musical?
Pink:   Where are you from?
Red:    Chicago
```

Transcript 1: intention confusion

Typing inefficiency – most people type much slower than they talk, which results in a much slower rate of communication. The average active chat user might take 30 seconds to enter a turn. This delay is experienced by the receiver as a silence in the conversation [2] – something that is established as being socially awkward and easily misunderstood [17]. To prevent or mask this delay, people tend not to wait for a response before starting their next turn. This tends to diminish the effectiveness of the slower typists and can lead to confusion. In transcript 2, we see Black and Pink (both fast typists) go ahead with another thread without waiting for Red to catch up. As a result, Red's comments are ambiguous.

```
Black:  Did you see that new Mel Gibson movie – I
        think it is called "Payback"?
Pink:   I saw the academy awards last night.  Did
        you watch it?
Pink:   yep.
Pink:   It was very violent, but funny.
Black:  You saw it?  You liked it?
Black:  How did it end up – who won?
Red:    I heard it was good.
Pink:   It was OK.  At least Titanic didn't win
        everything.
Black:  I guess you can only be king of the world
        once.
```

Transcript 2: thread confusion

Lack of status information – there is often no way to tell if a user is actively participating in a conversation or has wandered off from the computer all together. This often results in out of order turns, where someone asks a question again in a different way, not realizing that another user has been composing an answer for the first question. (See transcript 3).

In this transcript, 10 seconds had passed after Black asked his first question. Black assumed that this inactivity meant that Red was not interested, so he asked another question. However, Red, who is a slower typist, was actually typing his response the entire time. Now it is unclear to Black what question Red's answer is in response to.

```
Black:  Did you want to go skiing tomorrow?
Black:  Or maybe we should just go to dinner?
Red:    That sounds good – count me in.  I think
        that Blue will want to go as well.
```

Transcript 3: inactivity-related confusion

Lack of context – there is no way to tell what has gone on before in a conversation or what people are talking about now. This is true in a normal spoken conversation as well, but is a particular problem in chat because of the number of people constantly entering the conversations and the expense of using additional turns to bring new users up to speed.

High signal-to-noise ratio – since most of the participants in chat do not know each other, there is a great deal of introductory socialization phrases in any chat conversation. In addition, due to some of the previous factors mentioned, a large percentage of chat turns are corrections, clarifications, or repeats of previous turns.

General uselessness of the chat history – although most chat interfaces save the entire history of a particular session, it is rarely used in the chat context. For example, once an answer to a question has scrolled off the screen, users are far more likely to ask the question again rather than scroll back into the history. Part of the problem is the real-time nature of the chat – when you scroll back, you miss out on new things that are said. But even when the history has been saved for off-line review, most people still find it extremely ineffective to use. The out-of-order turns and high signal-to-noise ratio make it almost impossible to read.

With all of these difficulties, and with chat having essentially the same user interface it had from its earliest console days, we felt the time was ripe to experiment with some alternative chat interfaces.

VIRTUAL WORLDS TECHNOLOGY

In prototyping different chat UIs, we were aided tremendously by the Virtual Worlds platform, created within our group at Microsoft Research. The Virtual Worlds platform [16] is COM-based technology that allows programmers to quickly and easily create multi-user applications.

Virtual Worlds provides a persistent, distributed dynamic object system. Objects can be created and modified on the fly, including adding and removing properties and methods. The system takes care of managing client connections and replicating data between the server and the clients.

Clients can be created using any COM-compliant programming language, including DHTML, Visual Basic, and C++. The combination of DHTML and the Virtual Worlds platform makes an excellent rapid prototyping environment for multi-user applications.

More information on the platform, including downloading directions, can be found on the World Wide Web at http://research.microsoft.com/vwg/

THE “CONFERENCE” USER STUDY

We began our project by getting some data on the current chat interface, in order to have a baseline on which to compare alternative representations. We used a standard user study we call the “conference.” It has been used in the past to compare such things as teleconferencing, video conferencing, and face-to-face meetings.

In this study, a group of four to six people are asked to participate in a pair of on-line conferences. In the first conference, the groups would have to agree upon 5 videos to rent for a weekend retreat. In the second, they would decide which 5 albums to bring along on a long road trip. Typically, the two conferences are done using different technology (such as teleconferencing vs. face-to-face) in order to compare the differences.

Each conference lasts approximately 30 minutes, after which time the users are asked to complete a short questionnaire. The questionnaire elicits factual questions about the conference (such as, what types of music the user believe each of the other participants enjoyed), as well as qualitative questions (such as how they feel the communication flowed.) In addition to the surveys, the users may be videotaped for later analysis.

ATTACKING THE PROBLEM: STATUS

Most of the user interface improvements in existing commercial chat software are related towards improving the handling of multiple threads. This includes such things as giving users the ability to carry on multiple private conversations [MS-CHAT], or allowing them to indicate precisely which person or which turn they are responding to [V-CHAT, USENET].

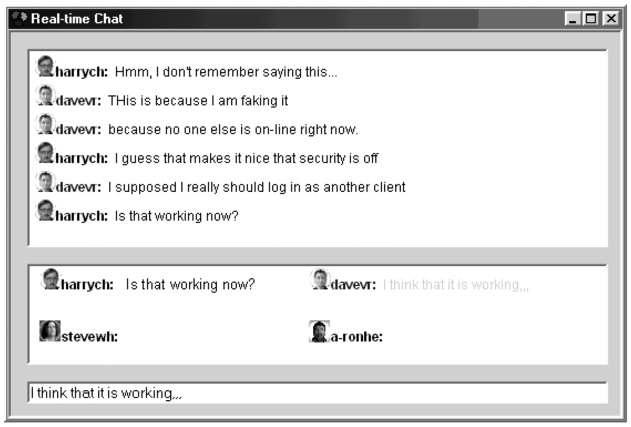
While we agree that it is worthwhile to explore the problem of thread management, we have for this study focused our energies on trying to understand and eliminate the initial conditions that lead to the multiple overlapping threads in the first place.

In spoken conversation, people do not engage in more than one line of conversation at the same time. We have social conventions against interrupting, and people are wait for other people to respond before speaking again. Silence in conversation is something we are conditioned to avoid, and spoken conversation has far fewer patches of “dead air” than exact same conversation typed.[17]

This led us to believe that we might eliminate many of the problems with threading by giving the users something to do after they complete a turn other than starting another turn, that did not feel like the equivalent of silence. We also suspected that if users where aware of what other participants were doing, they would be less inclined to enter turns that were simple status requests. These two suspicions led to the design of the “status client.”

STATUS CLIENT

We designed a prototype of an interface that showed the status of each user, as determined by keyboard activity.



harrych: Hmm, I don't remember saying this...
davevr: THis is because I am faking it
davevr: because no one else is on-line right now.
harrych: I guess that makes it nice that security is off
davevr: I supposed I really should log in as another client
harrych: Is that working now?

harrych: Is that working now?
davevr: I think that it is working,,,
stevewh:
a-ronhe:

I think that it is working,,,

Figure 2: Status Client

This status client keeps the same basic three-pane layout of a standard text chat client but augments it with several type of continuous status information. These are:

1) *Keyboard activity fader* – a user's name in the user list would highlight whenever a key was pressed, then fade back over 5 seconds to the original level
2) *Last line indicator* – a user's last turn would appear next to their name in the user list
3) *Intermediate text indicator* – each time the user pressed a key while composing their turn, the new text would appear next to their name in the user list. This text would be in a different color than the regular last line color to indicate its preliminary nature.
4) *Last line fader* – along with 2 or 3, the last turn text would fade out over 10 seconds.

Our prototype was able to activate each of these features individually, with the exception of 4), which required either 2 or 3 to be enabled. The prototype was coded using DHTML pages with VBScript on top of the virtual worlds system. For our experiment, we ran with options 2, 3, and 4 enabled. In this configuration, users would see every character every user typed as they were typing it, next to that user's name. When the user pressed RETURN, their text would be added as a new turn to the scrolling history, and would gradually fade from the user list.

STATUS CLIENT TESTING AND RESULT

For our first test, we ran two five-person groups (A and B) through the conference study. Group A used the client in standard mode with no status features for their first conference, and with status enabled for their second conference. This sequence was reversed for Group B. Rather than videotape the participants, we used client-side logging of the conversation. Each keystroke, mouse action, and incoming event was time-stamped and logged to a text file. Most of our analysis centered on these files. The participants were all experienced chatters, and as a group could type quite fast, averaging 250 CPM.

From a qualitative point of view, the users liked the status client, and all but one preferred it to the standard interface. Most people stated that they liked to watch other people type, but also admitted feeling embarrassed by their own typing skill being on display. Most users said they focused all of their attention on the user list with its live status, and paid little attention to the text history.

We were looking at several factors in the log files. The first was a quantitative analysis of the number of correctional turns of a percentage of total turns. We tracked this information by user as well as by conference. Our suspicion was that in the status prototype, there would be fewer corrective turns, as people would have more context over what other people were responding to. We were not, however, able to ascertain a difference in our study, as there were almost no turns dedicated to correction in either UI. (we will discuss this further in the conclusions).

The status client did show fewer out-of-order turns in the history. User would start to type an answer a question before the asker pressed RETURN, but they would wait to press RETURN themselves until the question was in the history. This resulted in questions being directly followed by their answers in the history, and made the history subjectively more readable.

We also wanted to see the affect on the total number of turns. Our suspicion was that in the status-enabled client, there would be fewer actual turns, as people would see that other users were typing and would wait for them to finish before starting another turn. However, the data showed that there was no significant impact on the number of turns. Regardless of interface, both groups had fewer turns on their second round than their first.

We were also looking for other artifacts from the real-time nature of the status information, such as would people be more or less likely to participate. Here we saw something quite interesting that we did not anticipate: when their keystrokes were being sent over the line as entered, users' increased both the speed and accuracy of their typing. We tracked the typing speed in uncorrected characters per minute, and the accuracy of the typists as a ratio of the number of keystrokes needed to generate a turn of a given length. With the standard client, our users averaged 250CPM at 75% efficiency. With the status client, speed increased to 280CPM at 85% efficiency. Our suspicion is that this was due to the feeling of being "on display", and might well go away with time as the novelty of the system goes away.

In general, the increase in typing speed and accuracy did not affect our users' ability to participate. However, one user (Red) had only marginal typing skills. With the standard client, Red typed 180 CPM with 64% accuracy – much worse than the other participants. With the status client, Red increased to be on par with the other users (224 CPM with 85% accuracy). But this accuracy came at a cost: he contributed less than 5% of the turns in the conversation (versus 20% for the average user), and his turns were only 15 characters long (vs. 30 for the average user). A follow-up interview confirmed that Red was concentrating so intently on his typing that he limited himself to only the occasional short turn.

With both interfaces, we noticed that people did not use the history much, if at all. They would not scroll back in the history to find a piece of information that was located off the screen. In fact, 10% of the turns were repeats and restatements of previous turns that were off the screen. In all four rounds, not a single person used the scroll bar during the tests. Many people suggested that it would be helpful to have a whiteboard or other persistent affordance during the conversation, similar to what one might find in NetMeeting or other CSCW software.

Our suspicion was that people would have an easier time answering the questionnaire with status client, due to the history being more readable. However, this turned out not to be the case. There was no significant difference between clients in the accuracy of the survey. Most users complained that it was too hard to find things a particular user said and suggested it would be nice to be able to filter the history by user.

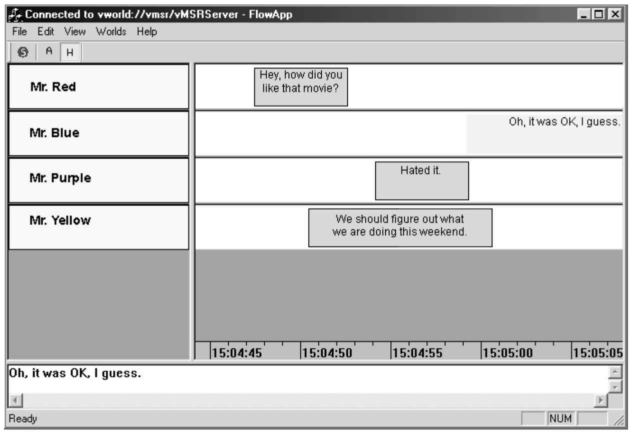
DESIGN AND REDESIGN OF THE FLOW CLIENT

One of the problems immediately apparent in the status client was that a great deal of the time sequencing information was lost in the history display. For instance, if you were watching the screen, you could tell that someone else had started typing after you did, but if they hit return first, their turn would still come before your own in the history. This lead us to suspect that perhaps the display of turns in a linear fashion based on transmission time might not be the best representation.

Based upon this observation as well as the qualitative and quantitative feedback of the status client, we began to design a new client. Our goal was to make something that would have the real-time benefits of the status client but that would preserve the temporal ordering of turns and provide easier matching to the speaker.

Our fundamental realization was that chat is closer to a streaming media type than anything else, and that it might be efficient to view it with a more traditional streaming media interface – the multi-channel timeline. These timelines are seen in many streaming media editing products, such as Macromedia's Director, Adobe Premier, and MIDI sequencing programs. In these interfaces, time flows from right to left, and the screen is broken up into a number of horizontal channels for different event sources, such as different MIDI channels, sprites, or film files.

Our first take on this can be seen in figure 3 – the Flow Client. The Flow client is an MFC application written in C++, using the virtual worlds technology for its client/server architecture.



Hey, how did you like that movie?

Oh, it was OK, I guess.

Hated it.

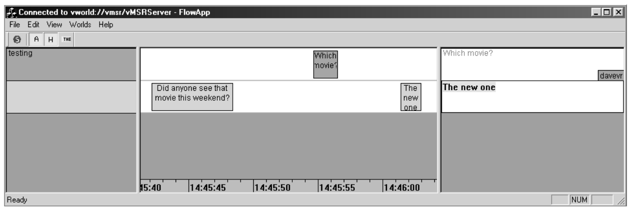
We should figure out what we are doing this weekend.

Oh, it was OK, I guess.

Figure 3: Flow client version 1

In this first version, we can see the user list on the left. Each user has a track on the screen. Time is constantly flowing, and the timeline display is constantly scrolling from the right edge of the screen (which is NOW) towards the left, at approximately 1 inch every five seconds. As a user begins typing, a colored box begins to stretch out in their timeline. The left edge of the box is at the time where they started typing. The right edge is set at the time they press return. In the meantime, the right edge stays aligned to the right side of the screen, and the text in the box is continuously updated with what the user has typed as they type it. If the user becomes inactive, the yellow box begins to fade until they continue typing. At any time, a user can mousedown and drag the timelines to peek back a bit in time. When they released the button, time would snap back to the present. We termed this "snap-back scrolling." In addition, the user could click a button to stop the time flow and scroll at their leisure.

Preliminary user tests revealed that people liked the client in general, but were bothered by the names being on the left while the new text was entered on the right. They also were confused by having to enter text down below while it showed up up above. Users also reiterated their request for some sort of persistent information in the UI.


Connected to vworld://vmsr/vMSRServer - FlowApp
File Edit View Worlds Help
testing
davevr
45:40
14:45:45
14:45:50
14:45:55
14:46:00
Ready
NUM
Which movie?

Did anyone see that movie this weekend?

The new one

Which movie?

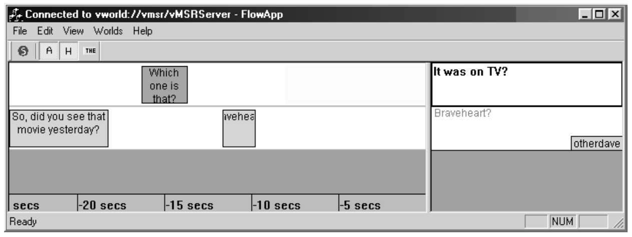
The new one

Figure 4: Flow client version 2

These comments lead to the design of Flow Client 2 (figure 4). In this version, the screen is divided horizontally into three sections. On the rightmost is status information that is out of the timeline. This is basically a single line of text that each user can set. In the middle is the timeline view, and at the right we see the user list. As a user types, the yellow box stretches out into the timeline as before, but the text is placed next to their name in the user list instead of in the yellow box. The text appears in blue to indicate that it has not been committed. If you paused while typing, your text would slowly start to fade to a lighter blue. Your own typing is done directly into the user list, and the text entry frame on the bottom has been eliminated. When the user commits the turn by pressing RETURN, the text is placed into the timeline rectangle and the text next to their name changes to black. This is similar to the status client.

We tested in front of users again. They liked the new way the user list worked, but did not like the persistent information area on the far left, saying it was "hard to read" and "confusing". Users were also bothered by the scrolling of the timeline ruler at the bottom of the screen. They felt that it was distracting and did not tell them what they really wanted to know, which was how long ago a particular turn took place.

This input lead to the third and final redesign:


Connected to vworld://vmsr/vMSRServer - FlowApp
File Edit View Worlds Help
A H THE
secs
-20 secs
-15 secs
-10 secs
-5 secs
Ready
NUM

Which one is that?

So, did you see that movie yesterday?

avehea

It was on TV?

Braveheart?

otherdave

Figure 5: Final Flow Client

In the final version (Figure 5), we eliminated the left-hand pane, and we changed the timeline to be a relative time indicator from "now" back to –30 or so seconds (depending upon window size).

FLOW CLIENT TESTING

Once we had a design that was deemed acceptable based on preliminary qualitative feedback, we conducted formal user studies.

In testing the previous interfaces, many of the users gave feedback that the conference room study was not representative of a "typical" chat experience. This was because it had a set topic to discuss, rather than the free-form topic-a-minute nature of normal chat. Indeed, in analyzing the data, there were very few off-topic turns in comparison to normal chat – only 3%.

Therefore, in the flow client tests, we attempted to more closely simulate a typical active chat room. We first did some less formal tests with two groups of five users from the status client tests. We then later tested five groups of four subjects in a more controlled, videotaped environment. Each group participated in two back-to-back chat sessions, once with the flow client and once with the standard client. The chatters were instructed to enter the chat session and get to know a few things about the other participants. Suggestions were given for conversation topics, but no particular topic or goals were mandated. In addition to the study participants, each chat session also had two "confederate" chatters who were instructed to act as study participants and keep the conversation flowing. The confederates were sometimes asked to bring up a particular subject (such as a knock-knock joke) that could be referred to later to examine the history usage.

RESULTS

Quantitatively, we expected the flow client to share many of the characteristics of the status client with regard to typing speed and accuracy. This was borne out by the data from users who had been tested with all three clients. In general, people typed more accurately and no more slowly when their typing was on display. Total number of turns, % of conversation per user, and average turn length were all similar between the clients.

We suspected that users would find the history easier to use and would consult it more readily than in other clients. The history was indeed consulted more, as every participant used the snap-back scrolling feature many times. The most-often stated reason for using it was to catch snippets of conversation that had left the screen. The number of turns dedicated to repeating information dropped from 5% in the standard client to less than 1%.

Despite this, many disliked the presentation of the history in the Flow Client. People felt that the UI was unfamiliar and preferred a vertical scrolling format. They felt the conversation left the screen too quickly and that all of the live typing areas were too hard to monitor. People were less accurate in completing a task that involved browsing the history, and felt more anxious with the Flow client in general. A similar complaint was the feeling that the flow client, with its large gaps between subsequent turns, made inefficient use of screen space compared to normal chat. This perception is interesting because an analysis of the conversation shows that in both cases, the flow client is actually slightly more efficient.

With four or fewer people in the room, a normal chat interface can potentially provide an efficient use of space. That is, each user can have four or more turns on the screen at once. We have seen that in chats with fewer participants, turns tend to be longer, which also increases the efficiency of the standard display of text history. Figure 6a shows a graphic representation of the same chat in a standard client and the flow client. Note that the flow client shows three to four turns for each user, while the standard client can display six of more.

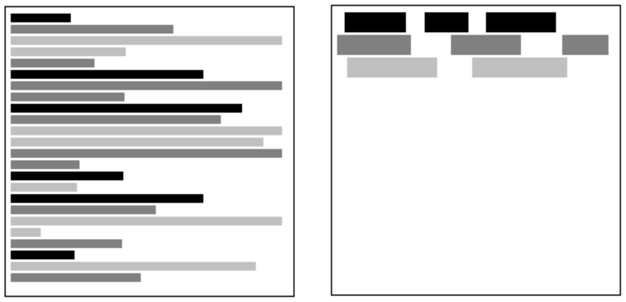


figure 6a: standard UI (left) and flow UI (right) displaying 3 people in chat

As more members join the chat, the flow client continues to display three or four turns per user, while the standard client becomes less and less efficient (Figure 6b).

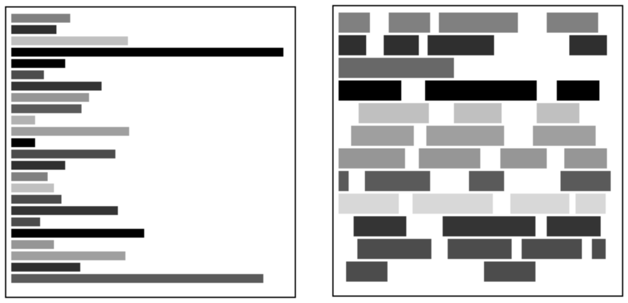


Figure 6b: standard UI (left) and flow UI(right) displaying 12 people in chat

With six or more speakers, turns in the standard client start to scroll off the screen very quickly, and the number of turns between adjacency pairs expands. This requires that more turns be dedicated to clarification and correction. In the flow client, the scroll rate remains constant regardless of the number of speakers.

CONCLUSIONS

Our usability results revealed some design problems in the application:

1) Continuous scrolling – a substantial problem users found with the client was with the large continuously scrolling area. Several users complained of a vertigo-inducing effect from having 75% of their field of vision scrolling slowly to the left. In the future, we hope to experiment with implementations that do not use smooth scrolling.
2) Too Much Activity - Users were confused and distracted by the amount of activity on the screen.
3) Novelty – as with any new interface, we faced resistance from users familiar with the established interface.
4) Flicker Problems – On certain machines, the text boxes would flicker as the users entered text. The flicker was extremely distracting to the users, especially when a half-dozen people were typing at the same time. (Affected users suggested displaying an epilepsy warning when the application starts.) This is a technical problem that we are hoping to correct in a future version, as several users mentioned it as the primary reason for disliking the Flow client.

FUTURE WORK

Despite the problems with our initial design, we feel that streaming media representations of chat are worth further investigation.

We are hoping to set up a chat site where we can conduct longitudinal studies of different chat clients. We feel many of the concepts introduced require some time for users to adapt to them before we can get valid statistical results. This seems to be particularly the case with transmitting each character as it is typed. We hope to have such a site up and running during the Fall of 1999.

One of the problems of busy public chat rooms has been the lack of context for new people entering the chat. We have tried experimental interfaces that preserve that last hundred or so turns of chat for a user when they connect, but the usefulness of this has been hampered by the overarching uselessness of chat histories in general. We suspect that with a variation of the flow interface, such a history would provide added value and context, and we hope to test this soon.

We are particularly interested in user interfaces for indicating the status of the user (typing, reading, away from the computer, etc.) We are investigating ways of doing this that do not require sending every character across the line.

Finally, we want to explore the ability to show chat overviews. We suspect that the flow client's display readily lends itself to a symbolic overview that can provide information (such as, which people talking and how often). We hope to add some overview capability into the next version of the client.

ACKNOWLEDGEMENTS

We would like to thank Shelly Farnham for her assistance in running the Flow Client user studies, and the Microsoft Usability Lab.

REFERENCES

1. Ackerman, M., Starr, B., "Social Activity Indicators for Groupware", Computer, June 1996, 37-42
2. Altun, A., "Interaction Management Strategies on IRC and Virtual Chat Rooms", SITE 98 proceedings, March 10-14, 1998, 1223-1227
3. Beaudouin-Lafon, M., Karsenty, A., "Transparency and Awareness in Real-Time Groupware Systems", UIST proceedings, 1992, 171-180
4. Bowers, J., Pycock, J., O'Brien, J., "Talk and Embodiment in Collaborative Virtual Environments", CHI 1996 proceedings, 55-65
5. Dourish, P., Bellotti, V., "Awareness and Coordination in Shared Workspaces", CSCW proceedings, 1992, 107-114
6. Dourish, P., Bly, S., "Portholes: Supporting Awareness in a Distributed Work Group", CHI proceedings, 1992, 541-547
7. Dringus, L., Adams, P. (chair), "A Study of Delayed-Time and Real-Time Text-Based Computer-Mediated Communication Systems on Group Decision-Making Performance.", Dissertation, Nova University, 1991, 217 pp.
8. Ellis, C., Gibbs, S., and Rein, G., "Groupware: Some Issues and Experiences", Communications of the ACM, Vol. 34, No. 1, 1991, 39-58
9. Garcia, A., Jacobs, J., "The Interactional Organization of Computer Mediated Communication in the College Classroom", Qualitative Sociology, Vol. 21, No. 3, 1998, 299-317

10. Murphy, K., Collins, M., “Communication Conventions in Instructional Electronic Chats”, First Monday, Vol. 2, No. 11, 1997
11. Powers, S., Mitchell, J., “Student Perceptions and Performance in a Virtual Classroom Environment”, American Educational Research Association Annual Meeting, March 1997
12. Reid, E., “Electronic Chat: Social Issues on Internet Relay Chat”, Media Information Australia, 67, 1993, 62-79
13. Rintel, S., Pittam, J., “Strangers in a Strange Land: Interaction Management on Internet Relay Chat”, Human Communication Research, Vol. 23, No. 4, 1997, 507-534
14. Sproul, L., Keisler, S., “Reducing Social Context Cues: Electronic Mail in Organizational Communication”, Management Science, Vol. 32, No. 11, 1986, 1492-1512
15. Tang, J., Issacs, E., and Rua, M., “Supporting Distributed Groups with a Montage of Lightweight Interactions”, CSCW proceedings, 1994, 23-34
16. Vellon, M., K. Marple, D. Mitchell, and S. Drucker. “The Architecture of a Distributed Virtual Worlds System.” Proc. of the 4th Conference on Object-Oriented Technologies and Systems (COOTS). April, 1998.
17. Wiemann, J., Knapp, M., “Turn-Taking in Conversations”, Journal of Communication, 25, 1975, 75-92

mIRC. By Khaled Mardam-Bey. Shareware software available at http://mirc.stealth.net
MS-Chat: by Microsoft, Inc. Free download available from http://www.microsoft.com
v-Chat: by Microsoft, Inc. Free download available from http://www.microsoft.com