Semantic Telepointers for Groupware

Saul Greenberg, Carl Gutwin and Mark Roseman
Department of Computer Science, University of Calgary
Calgary, Alberta, Canada T2N 1N4
phone: +1 403 220 6015
email: {saul,gutwin,roseman}@cpsc.ucalgary.ca

Abstract

Real time groupware systems often display telepointers (multiple cursors) of all participants in the shared visual workspace. Through the simple mechanism of telepointers, participants can communicate their location, movement, and probable focus of attention within the document, and can gesture over the shared view. Yet telepointers can be improved. First, they can be applied to groupware where people's view of the work surface differs—through viewport, object placement, or representation variation—by mapping telepointers to the underlying objects rather than to Cartesian coordinates. Second, telepointers can be overloaded with semantic information to provide participants a stronger sense of awareness of what is going on, with little consumption of screen real estate.

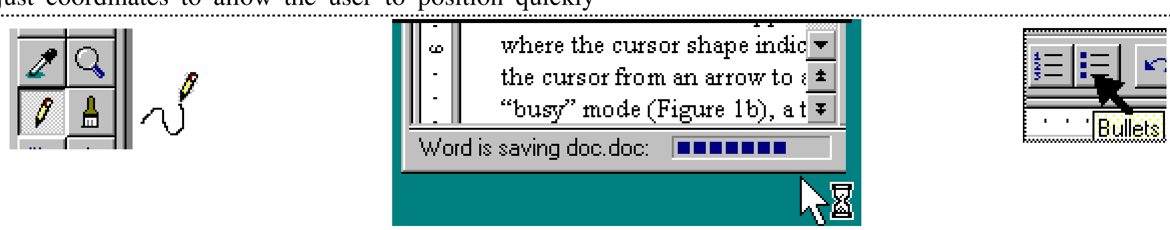
1. An Historical Introduction

Twenty-five years ago, Douglas Engelbart demonstrated an interface that tied a pointing device (a mouse) to a "tracking spot" (a very small cursor) that was always on display [1,2]. Today, virtually all modern graphical user interfaces incorporate similar pointing devices and cursors. Their basic functionality remains unchanged: the cursor position on the window follows the relative or absolute Cartesian coordinates of the pointing device, although variable speed algorithms sometimes adjust coordinates to allow the user to position quickly and accurately a cursor on large displays.

The only real additions to Engelbart's tracking spot are the limited ways cursors are overloaded to reflect semantic information about the underlying application or system state. One typical use of such a *semantic cursor* is to have its shape reflect a user's interaction mode. For example, Figure 1a shows Microsoft Paint, and the pencil cursor indicates that the user is in "sketch mode". A second use reflects system state, illustrated in Figure 1b by the hourglass cursor that redundantly reminds the user that Microsoft Word is busy saving a file. Semantic cursors can also provide help information, such as brief descriptions of objects that are being passed over. Examples are Apple's balloon help and most Microsoft toolbars (Figure 1c): while not actually part of the cursor, pop-up help is activated by the cursor, and the description is close enough to it that the user considers them a unit.

Engelbart also demonstrated how cursors can work in groupware when he and his distant partner shared the display of a terminal [1,2]. Both had a mouse, and individuals could see on the display one's own tracking spot as well as the one belonging to the other person. Obvious in the film of Engelbart's demonstration is that these *multiple cursors* or *telepointers* help mediate conversation. Their location indicates where the other person is in the workspace; they focus attention around what people are doing; and they are used to gesture around the shared objects on the display [2].

Engelbart was well ahead of his time with telepointers,



a) Mode cursor showing the mode selected from the paint pallette

b) A system state cursor, showing that the application is busy and that the user must wait.

c) Help cursor, showing information about the icon under the cursor

Figure 1. Conventional uses of cursors to show semantic information

and only a few point systems used them over the years, e.g., multiple telepointers in Mblink [13], and the single active cursor in shared window systems [3]. It wasn't until 1987 that telepointers (and indeed groupware in general) were revisited with any seriousness. CoLab was a computer supported meeting room with a variety of electronic meeting tools [14,15]. It included a shared virtual whiteboard, and participants could select pens for drawing, erasers for erasing, and a single large telepointer for pointing at objects in the display. Its developers discussed design issues behind telepointers, and came up with some interesting (and sometimes wrong) recommendations from their experiences.

In 1989, the value of telepointers was placed on an empirical footing when Tang developed a framework cataloging activities in a shared drawing space [16]. In his study of face to face small group design meetings, participants using a shared drawing space (white-boards or large sheets of paper on a table) produced three significant actions: *listing* of text, *drawing* of images, and *gesturing* through hand motions. He mapped these actions to their purposes: storing information, expressing ideas, and mediating interaction. He counted the distribution of these actions and functions, and found that gestures comprised ~35% of all actions, and were used by the group mostly to mediate interactions and to express ideas. The implication to groupware is that shared distributed work surfaces must support gestural expression. Consequently, developers of video systems used video overlays to show other peoples' hands over the images [10], while those interested in computational workspaces used multiple telepointers as gesturing surrogates [6,9].

We have now come full circle. Although we have a better understanding of why telepointers are valuable, their appearance in commercial groupware has evolved little from Engelbart's original demonstration. In this paper, we will show how telepointers can be improved in two different ways. First, we can apply them to groupware where people's view of the work surface differs—through viewport, object placement, or representation variation—by mapping telepointers to the underlying objects rather than to Cartesian coordinates. Second, we can overload telepointers with semantic information to provide participants with a stronger sense of awareness of what is going on. These extensions demand that multiple cursors know something about the underlying work surface and application semantics, which is why we call them "semantic telepointers".

2. Telepointers for Relaxed-Wysiwis Views

Conventional real-time groupware provides participants with strict "what you see is what I see" (wysiwis) views, where participants see exactly the same thing across their displays [14]. However, this has proven limiting, and recent groupware has relaxed strict view-sharing in three ways: a) people can now have different viewports into a workspace; b) objects in the workspace can be formatted differently across displays; and c) objects can have differing representations [14]. The idea is that relaxed-wysiwis view-sharing makes groupware more flexible and better able to match particular needs of participants and the way they actually work, especially in large workspaces that contain many artifacts. In this section, we will show how we can map telepointers to the differing views in these three styles of relaxed-wysiwis views by using progressively more knowledge of the application semantics. Because these techniques introduce unsolved usability issues, we list them as well.

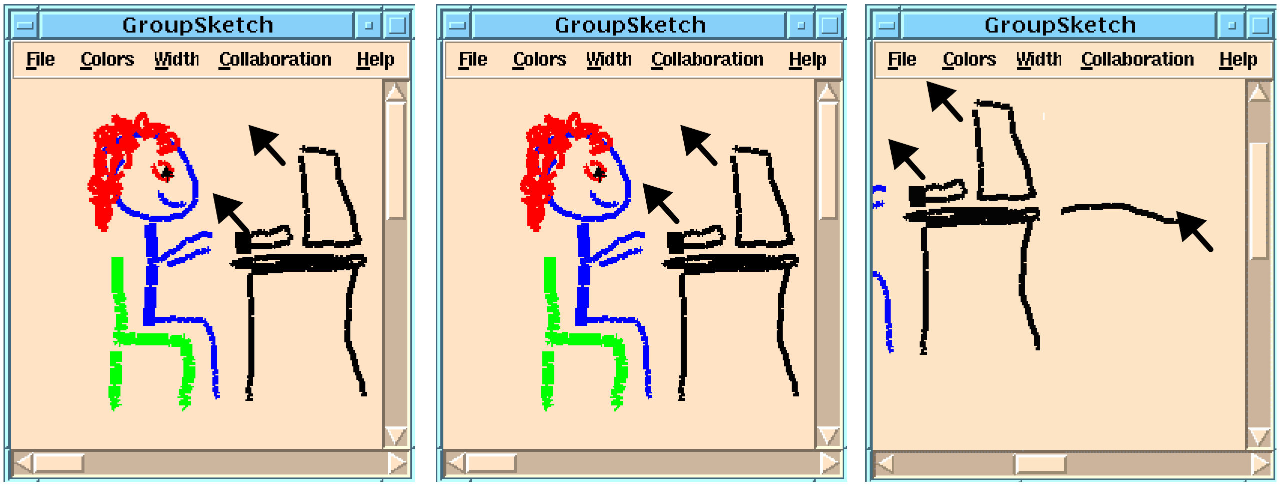
2.1. Viewport Differences Between Views

Telepointers are easy when views are identical, for cursors are just mapped to the individual displays at their respective (x,y) window coordinate positions. For example, Figures 2a+b show matching views of two people, Saul and Carl, and the cursors are in the same relative position on the window. If we relax wysiwis by allowing people to scroll independently, their viewports will differ and cursor locations must be positioned relative to the underlying work surface rather than the window. For example, when Carl scrolls his view (Figure 2c), the cursors are positioned correctly on the drawing rather than the window (the 3rd cursor belongs to yet another person).

This mapping of telepointers to the viewport is typically implemented in two ways. First, if cursors are drawn in the application area, we just position them using the world coordinate space. Second, if cursors are drawn on a separate layer independent of the application, we now have to map the relative window coordinates of the cursor layer to the world coordinates of the application.

Unsolved issues. Differing viewports introduce several problems at the interface level.

1. Another person's cursor will not be visible if it is outside the viewport. For example, in Carl's view (Figure 2c) a third person's cursor is visible, which is invisible in Saul's view (2a). Consequently it is hard for Saul to gauge where that person is working and how active they are. Awareness between participants must be promoted by alternate strategies [8].
2. The verbal talk accompanying gestures can be confusing when deixis reference is uncertain. For example, the phrase "what about this thing here?" is unambiguous when it refers to the object being pointed to. However, the phrase "what about this thing here, at the top of the window?" is ambiguous, as it now describes both the gestural act and the (perhaps) incorrect relative location of the object.



a) Saul's view b) Carl's view before scrolling c) Carl's view after scrolling

Figure 2: Cursor actions when viewports differ

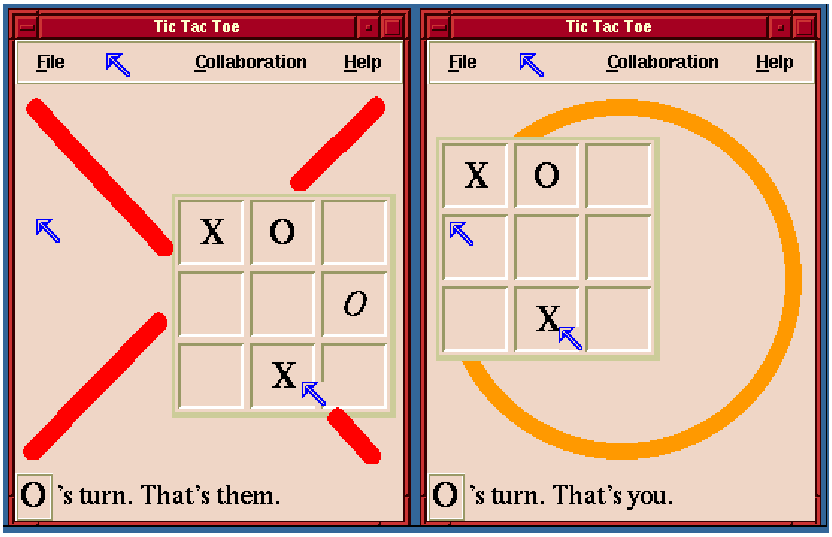
2.2. Format Differences Between Views

The second relaxed-wysiwis style allow views to be formatted differently, where objects that are otherwise identical appear in the views in different locations. This is appropriate when participant's windows are different sizes, requiring the application to reformat its objects to fit the space, such as in text displays. Alternatively, the groupware application could contain components whose position in the window may be customizable by individuals. In both cases, telepointers can still be implemented correctly by mapping their position to the underlying application objects, rather than a literal coordinate system.

Our first example illustrates telepointers in automatically reformatted displays. GroupWeb is a groupware version of a world wide web browser that allows a group to share views of HTML pages [5]. In GroupWeb, participants can resize their windows independently (Figures 3a+b), and the underlying text is reformatted accordingly. As a person moves their telepointer over a word, its remote counterparts appear correctly on top of the same word on the other displays, e.g., the word 'Web' in 3a+b, even though the word's actual positions in the windows differ radically. To implement this, GroupWeb telepointers are mapped to character objects. As a person moves their telepointer, the underlying character and its position are discovered. That position as well as the telepointer coordinates relative to

a) Saul's view b) Mark's view

Figure 3: GroupWeb, showing two people's views in windows of different sizes



a) Saul's view (as X) b) Carl's view (as 0)

Figure 4: Groupware Tic Tac Toe, with different board locations

a) gesture is made here b) but appears badly

Figure 5: Stroke appearance

the character origin are transmitted, causing remote telepointers to appear over the corresponding characters. Images are done the same way: telepointers are also drawn relative to the origin of the image in the other views, independent of the image's actual position in the window.

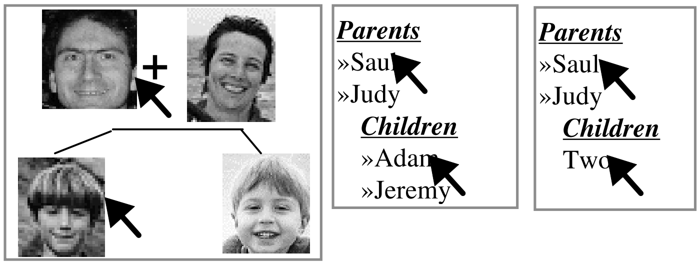
Our second example illustrates telepointers in groupware where an object's location in a particular view is customized by an individual. Figure 4 shows a groupware tic-tac-toe game, where participants can position the board anywhere they wish in the window. There are several objects here: the movable game board, the static menu bar on top, the labels at the bottom, and the background (containing the large 'X' and 'O'). Again, the cursor position is mapped relative to the location of the object. As seen in the differing views in Figure 5, telepointers will appear correctly on game boards that are at different locations (e.g., the pointer over the 'X' in the bottom row of the board), and over objects that are in static positions (e.g., the telepointer in the menu bar). The third anomalous telepointer will be discussed shortly.

Unsolved issues. Differing view formats between displays suffer the same usability problems as differing viewports, as well as several new ones.

1. Empty and interstitial space is problematic. When a telepointer moves onto a spot unoccupied by an underlying object, it is not clear how it should be mapped onto the other displays. Although we could use the coordinate space of the background, the spatial relations of objects to it are suspect and even misleading. Figure 4 illustrates that what is 'empty' on one display may be populated by an object on another's display. In 4a, Saul has moved his cursor to the mid-left of the window, an apparently empty spot. When the telepointer is drawn in the corresponding position in Carl's window (4b), it appears over the game board, leading to possible misinterpretation of Saul's gesture by Carl. While we could draw the telepointer relative to the last object passed over rather than the background, similar problems occur.
2. Because telepointers support gesturing, we must consider how a gestural stroke—the temporal movement of a telepointer—appears across displays with different formats. For example, consider the differing formats of the text string in Figure 5. The person in 5a circles the phrase "some text and". This gesture is drawn discontinuously on the display in 5b, because the word "and" is formatted onto a different line. The meaning of these gesture strokes could be lost in these kinds of situations.
3. Similarly, discontinuities between objects cause gesture movement to be jumpy. A smooth movement from one object to another on one person's display can appear as abrupt jumps on the other display when the objects are in quite different positions relative to each other. For example, if Carl smoothly moves the cursor in the menu bar to the top-left 'X' in the game board. (Figure 4b), Saul would see it (in Figure 4a) as moving downwards onto the empty space, and then suddenly jumping right to the X on the board.

2.3. Representational Differences Between Views

The final form of relaxed-wysiwis allows different views of objects to be generated from the same data model. For example, consider the family tree, shown in Figure 6a as a pictorial hierarchy, in 6b as an expanded indented list, and in 6c as a partially collapsed list. As with differing formats, telepointer positions can be approximated by attaching them to objects, as shown across the figures. In this case, however, mapping takes a further level of indirection: the system must find what view is underneath the telepointer, which data model the view is tied to, and what alternate views are being generated from the model on the other displays. The telepointer position must then be translated to those other representations. In many cases, a one to one mapping function may not be possible.



a) pictorial hierarchy b) indented list c) collapsed

Figure 6: Three representations of a family tree

Unsolved issues. Usability and implementation is increasingly problematic over the other styles of relaxed-wysiwis views.

1. There is no consistent notion of a coordinate space within an object, and it is unclear how we can represent exact telepointer positions and motions made over one view to its counterpart on other displays. One solution is simply to move a cursor to an object's center, and not show any internal motion at all. However, this removes the dynamics of a gestural stroke, and causes jumpy animation. Another solution is to translate the motion over one object into a stylized motion over its counterpart. For example, a horizontal stroke across a picture in Figure 6a would be translated to a horizontal stroke across the appropriate name in 6b+c. Perhaps gesture recognition can further identify stroke styles, so that the *meaning* of the stroke can be captured and displayed appropriately.
2. With differing representations, several objects may be collapsed into a cluster, or removed from the view altogether. In Figure 6c, for example, the nodes "Adam and Jeremy" are collapsed into the single node "Two". While it may be reasonable to tie cursor motion over both nodes to its collapsed counterpart, this could lead to possible confusion between participants. If the node was completely invisible, no obvious solution presents itself.
3. People's verbal talk across displays may not match the views being pointed to. For example, "Look at Judy's smile" said over view 6a has little meaning in 6b+c.

2.4. Discussion

This section has illustrated how telepointers can be applied across a spectrum of same view/different view displays. It shows that as relaxed-wysiwis increases, cursors and their movement can still be shown. The catch is that people's interpretation of the telepointer and its strokes becomes more problematic: a telepointer's location may be ambiguous; it may be hidden from view; telepointer tracking can be discontinuous; and people's gestures enacted as telepointer motions may be more difficult to interpret.

From an implementation perspective, the complexity of mapping telepointer positions between views increases with the degree of view differences, and more has to be known about the semantics of the application objects to do the mapping. Window coordinates give way to world coordinates when viewports differ; to object coordinates when formats differ; and to stylized object-specific coordinates when representation differ. While simple approaches to telepointers could be supplied by a groupware toolkit (as done in GroupKit [12]), at some point the semantic constraints will require the programmer to supply intermediate routines that map a telepointer's presence and motion over one object into some reasonable representation on the other display.

Of course, the approximate measures given by telepointers in relaxed-wysiwis may suffice for many situations. If people are mostly concentrating on their individual work, seeing other's approximate telepointer activity may suffice to give a sense of awareness of what is going on. When there is need for a tightly-coupled discussion, the participants will likely shift their views into something resembling a strict-wysiwis situation.

In summary, this section advanced traditional telepointers by demonstrating how they can be applied to views that differ between participants. The next section will show another way that telepointers can be improved by overloading them with information useful to participants.

3. Overloading Telepointers

Engelbart's early cursor was just a tracking spot, a dot moving across the display. As mentioned in the introduction, modern systems now overloaded cursors to show a variety of information, such as mode and system state, and even help messages. Groupware systems can also leverage overloaded cursors to show participants critical information about their electronic meeting, as first seen in more current systems such as BoardNoter [15]. This section explores several ways cursors can be overload, as well as the benefits supplied to groupware. We will show how telepointers can provide extra information to the group about participants' interaction modes, identity, actions, and gestures.

3.1. Mode and State Information

In single user systems, mode information is used to provide feedback on system state changes, and to remind people of what interaction state (mode) they are in. In groupware, the same kind of information can indicate what state others are in, and what they intend to do. This information is critical in a groupware setting. In particular,

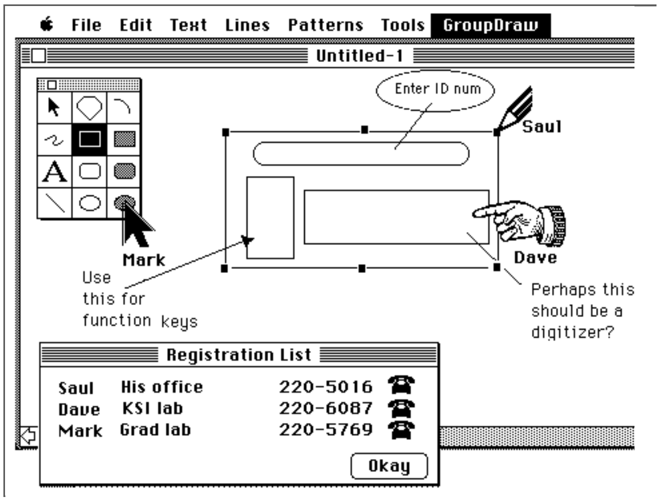


Figure 7: Overloaded telepointers in GroupDraw

it is quite easy for participants in a groupware conference to miss actions of others that trigger state changes. For example, a person's selection of a tool on a palette by a quick button press may be overlooked by others, while keystroke actions that change modes are invisible.

One solution is to overload individual telepointers to reflect mode information across all displays. Figure 7 gives an example, in this case from our GroupDraw application, a shared drawing tool [6]. Participants select tools and modes from the shared palette on the left, and the telepointer shapes reflect what mode people are in: Mark is in the selection mode (the pointer) and Saul can draw (the pencil). Knowing the mode hints at the intentions of others e.g., Saul will likely draw because he is in a drawing mode. Modes peculiar to groupware are also possible; for example, Dave is using a special "attentional" telepointer to grab the attention of others, invoked by pressing a modifier button.

Similarly, telepointers can inform people of system state. A common example in groupware is to show participants feedback of locked objects that are being used by one person and thus not available to others e.g., the Calliope multi-user text editor [11] changes a person's telepointer image to show a lock when a text fragment is being edited .

To implement these, application developers must be able to modify the cursor shape to indicate the relevant mode and state information. For example, shapes of individual telepointer can be altered simply by setting appropriate values in a cursor's bitmap field, as is currently done in single user systems.

3.2. Identity Information

Maintaining awareness of others in real time groupware is important [8]. This, of course, includes knowing who is in the workspace and where they are working. Telepointers are obviously important here, as they do indicate someone's presence and where they are focusing their attention. In our experiences, the speech that accompanies another person's actions during closely coupled collaborations will identify who owns the telepointer [6]. However, uncertainty of identity can arise during quiet periods, or in meetings with many people.

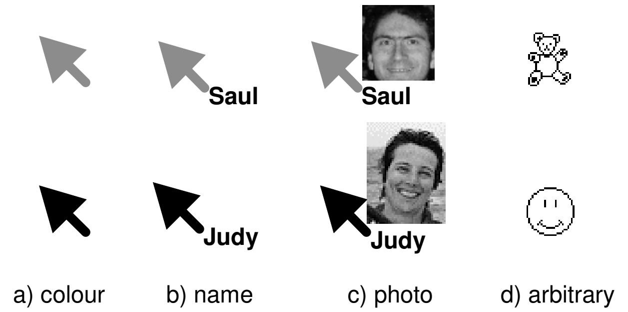


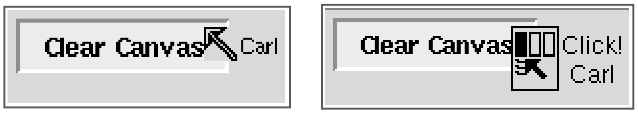
Figure 8. Various strategies for denoting identity

Telepointers can be overloaded to indicate the identity of their owners. For example, participants using applications built in GroupKit [12] declare a personal colour, and their telepointer is automatically drawn in that colour. While colour is not necessarily a good indicator of identity, it does serve to distinguish the multiple cursors. Better still, we can attach a name to the telepointer (Figure 8b) or even a photo (Figure 8c). The cost is increased clutter, and the risk of occluding objects behind the pointer. The Aspects groupware product uses another approach: people can choose arbitrary cursor shapes to identify them (e.g., Figure 8d). The problem is that shape can no longer be used to indicate mode changes, and participants must learn the (perhaps meaningless) associations between people and cursor shapes.

3.3. Action Information

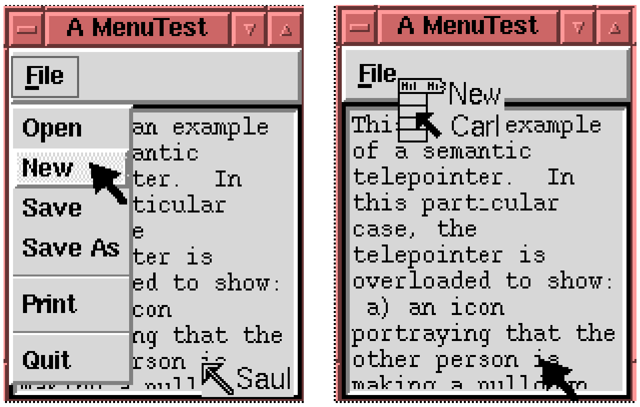
In our real world, people tend to control their environments through large actions that are easily visible to others. In groupware, seeing other's actions can be problematic. Small and rapid actions can be overlooked, as with the button press scenario mentioned earlier. Alternatively, other's actions may be too large or clumsy to show reasonably in the limited screen space available i.e., seeing another person pop up and navigate through a large, hierarchical menu. Overloaded telepointers can mitigate this problem, either by making easily missed actions larger, or by replacing large actions by a more condensed and stylized version of it.

Our first example, illustrated in Figure 9 considers the quick button press. In this case, the actions of the person pressing the button in Figure 9a is made larger to other participants by changing the telepointer for a modest length of time to show the mouse button being pressed,



a) Carl presses the button b) Saul's view of it

Figure 9: Action information in a button press



a) Carl selects an item b) Saul's view of it

Figure 10: Action information in a menu press

(Figure 9b). At the same time, a "click" sound is played to present additional cues. If the underlying button is also shown depressed, then the cursor just provides redundant feedback which acts to accentuate the action.

Our second example, in Figure 10, is slightly more complex. Here, Figure 10a shows Carl navigating through a pop-up menu. Showing the complete menu on Saul's display could be annoying, especially if he was working in the area immediately underneath it. Instead, Carl's telepointer image and labels are altered to indicate a menu selection is being made, and what item is being selected. In this case, the same semantic information of a menu action is shown on other displays concisely and with little or no loss of meaning.

3.4. Gestural Information

Because of performance limitations or problems with translating telepointer positions in relaxed-wysiwis settings, we may want to capture and present essential gestures in a stylized manner. In our previous work, Hayne, Pendergast and Greenberg [9] describe a few ways that gestures can be captured and transmitted.

The most basic gestural action is pointing. If telepointer movement cannot be transmitted easily, a person can indicate a "point" request, perhaps by pressing a mouse button (first implemented in Boardnoter [15]). When the message is received, the telepointer is moved to its new position over the object. Because these movements are discrete, we do not run into the problem of representing gestural strokes.

However, discrete moves can be easily missed. A "point and quiver" approach attracts the viewer's attention by modifying the simple move to make the telepointer jitter in its new position for a limited time.

In many cases, seeing a cursor move from one object to the next is valuable. Instead of trying to handle all telepointer movement points, the system can capture the motion as a discrete move from one object to another, and transmit only those points. The receiving station would be responsible for computing points in-between these two, and for animating the telepointer on that track. Similarly, we have already mentioned how stroke recognition could be used to translate a gesture from one person to a reasonable counterpart on the other display.

In the above situations, the system can send fewer messages down the wire, as a series of telepointer moves are collapsed into small messages that describe the semantics of the move rather than the literal coordinate stream. However, additional CPU processing is required to translate to and from the motion of the telepointer action and its semantic meaning.

Discussion and Summary

We believe telepointers are a necessary part of any real time groupware application that contains a shared visual space. They act as gestural surrogates, and are used by participants to express ideas and to mediate interaction. Their very presence in the workspace also provides awareness to others of who is present, where they are working, and what they are doing.

While telepointers have been around for a while, this paper reconsidered their role in groupware. First, groupware is shifting from strict to relaxed-wysiwis views, and this paper introduced some ways that telepointers can be managed in these settings; while there are unsolved issues, the approaches provide the next step. Second, telepointers are a natural focus of attention for group participants, and they can be leveraged to show information vital for smooth collaboration: interaction modes, system state, identity, actions of others, and so on. While overloaded cursors are not new, they have appeared only sporadically in today's groupware systems.

Telepointers are problematic for groupware implementers, as all window systems support only a single cursor. Most developers have to implement telepointers from scratch, which is no easy task. This is the likely reason why telepointers are not implemented in some groupware systems, or why they are still fairly crude.

In our own work, we have developed a groupware toolkit called GroupKit [12] that includes telepointers as a programmable widget. Application developers using GroupKit can include telepointers in a few lines of code. GroupKit currently maps telepointer location to underlying widgets, and they are partially tuned to obey the widget's semantics. For example, telepointers appear in the correct position in certain scrollable widgets, such as graphical canvases and text boxes, allowing people to have different viewports (as done in Figure 2). They also appear over the correct character in text widgets, which means text can be formatted differently. This was seen in GroupWeb (Figure 3), also built in GroupKit. Telepointers can also be tied to particular widgets that appear in different locations in the window, and the tic-tac-toe example in Figure 4 was done this way.

We are implementing a new version of telepointers to allow overloading. The current prototype constructs a telepointer with three glyphs: a bitmap and two labels. The API to the telepointer allows programmers to set the properties of these glyphs, which means that the cursors shown in Figures 7 through 10 can be easily constructed.

Telepointers can be used and enhanced in ways that go beyond this paper. For example, we have built telepointers that react to the size of the group [9]. When group size is small, telepointers are large. As more members enter, telepointers shrink progressively, eventually reaching the size of a few pixels. While the movements of these tiny cursors provide a sense of what is going on, it does so with minimal clutter. When people perform editing actions, cursors are enlarged to full size and include overloaded information such as identity and mode state.

As a second example, we have considered how telepointers can facilitate turn-taking [9]. Rather than supply everyone with a telepointer, a restricted pool of pointers are made available. Choosing one removes it from the pool, and transforms a person's single-user cursor into the telepointer. Strict turn-taking is implemented by making only one pointer available in the pool. Confusion and clutter that could occur in large meetings (e.g., remote presentations with 30 to 40 participants) can be minimized by making only several pointers available, thus limiting the number of people that can be active in a large meeting.

We recognize that 1-dimensional telepointers are crude approximations of our hand and body gestures. Video-based gesturing [10] is certainly richer, although technically limiting. We are also seeing virtual environments where people's hand and even body motions are captured and displayed as 3-d artifacts. In spite of these exciting possibilities, most of today's computer technology is still oriented to flat graphical displays and a mouse. Until this changes, the lowly telepointer can act as a reasonable and effective surrogate for gestures.

Acknowledgments. Alex Mitchell contributed ideas and code about telepointers. This research is supported by the Intel Research Council and NSERC.

Availability. GroupKit is available from: http://www.cpsc.ucalgary.ca/projects/grouplab/.

References

1. Engelbart D. and English W. (1968). A research center for augmenting human intellect. In *Proceedings of the Fall Joint Computing Conference* Volume 33, pp. 395-410, Montvale, NY, AFIPS Press. Reprinted in [7].
2. Engelbart, D. and English, W. (1994) A research center for augmenting human intellect. SIGGRAPH Video Review, 106, Videotape, ACM Press.
3. Greenberg S. (1990). Sharing views and interactions with single-user applications. In *Proceedings of the Conference on Office Information Systems*, pp. 227-237, ACM Press.
4. Greenberg S., Hayne S., and Rada R., eds (1995). *Groupware for real-time drawing: A designer's guide*, McGraw-Hill Europe.
5. Greenberg S. and Roseman M. (1996). GroupWeb: A WWW browser as real time groupware. In *Companion Proceedings of the ACM SIGCHI'96 Conference on Human Factors in Computing System*, pp. 271-272, ACM Press.
6. Greenberg S., Roseman M., Webster D., and Bohnet R. (1992). Human and technical factors of distributed group drawing tools. *Interacting with Computers*, **4**(1), pp. 364-392, December. Reprinted in [4].
7. Greif, I. (ed) (1988) *Computer-supported cooperative work: A book of readings*, Morgan-Kaufmann.
8. Gutwin C. and Greenberg S. (1996). Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation. *People and Computers XI (Proceedings of the HCI'96)*, A. Sasse, R.J. Cunningham, and R. Winder, Editors. Springer-Verlag.
9. Hayne S., Pendergast M., and Greenberg S. (1994). Implementing gesturing with cursors in group support systems. *J Management Information Systems*, **10**(3), pp. 43-61, Winter. Earlier version republished in [4].
10. Ishii H., Kobayashi M., and Grudin J. (1993). Integration of interpersonal space and shared workspace: Clearboard design and experiments. *ACM Transactions on Information Systems*. October. Reprinted in [4].
11. Mitchell, A. and Baecker, R.M. (1996). The Calliope multi-user shared editor. Department of Computer Science, University of Toronto, Toronto, Canada.
12. Roseman M. and Greenberg S. (1996). Building real time groupware with GroupKit, a groupware toolkit. *ACM Transactions on Computer Human Interaction*, **3**(1), pp. 66-106, March.
13. Sarin S. and Greif I. (1985). Computer-based real-time conferencing systems. *IEEE Computer*, **18**(10), pp. 33-45. Reprinted in [7].
14. Stefik M., Bobrow D. G., Foster G., Lanning S., and Tatar D. (1987). WYSIWIS revised: Early experiences with multiuser interfaces. *ACM Transactions on Office Information Systems*, **5**(2), pp. 147-167, April.
15. Stefik M., Foster G., Bobrow D., Kahn K., Lanning S., and Suchman L. (1987). Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Communications of the ACM*, **30**(1), pp. 32-47. Reprinted in [7].
16. Tang J. C. (1991). Findings from observational studies of collaborative work. International Journal of Man Machine Studies, 34(2), pp. 143-160, February.