Floor Control in a Highly Collaborative Co-Located Task

Brad A. Myers, Yu Shan A. Chuang, Marsha Tjandra, Mon-chu Chen, and Chun-Kwok Lee

Human Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
bam@cs.cmu.edu
http://www.cs.cmu.edu/~pebbles

ABSTRACT

"Floor control" is the protocol which determines which user has control and how to take turns when multiple people share a limited resource such as a single cursor in a synchronous task. First, we provide a new analysis and classification of floor control mechanisms. We then studied eight collaborative conditions in a highly-collaborative computer-based task, where all the subjects were co-located. We studied doing the task without a computer, compared to seven techniques where each user had an input device. This included two techniques where all users had their own cursor, and five floor control techniques for sharing one cursor. The floor control techniques included: having a moderator decide the turn, averaging all inputs together, blocking the other's input while the cursor was in use, explicit release, and explicit grab. We found no previous studies of all these mechanisms, although one prior paper predicted that the blocking mechanism should work best. Our primary result is that giving everyone a separate cursor works best, and we found no significant differences among the times using the floor-control mechanisms.

Keywords: Floor Control, CSCW, Meetings, Multiple Cursors, PDAs, Pebbles.

INTRODUCTION

"Floor control" refers to the management of interaction among participants in meetings. This comes from expressions such as "who has the floor" or "yielding in the floor" in formal meetings. Whenever there is a resource that must be shared among the participants, floor control issues arise. Rules of etiquette are typically sufficient for daily human-human interactions. However, mechanisms other than social protocol might be needed in computer-mediated group work.

In many synchronous (also called "same-time") computer-supported cooperative work applications, each user is given a separate cursor. However, when trying to share legacy applications, such as Microsoft Excel, Adobe Illustrator, or MacroMedia Director, it is technically difficult or impossible to have multiple cursors that can control the application. Sharing of these kinds of applications is frequently necessary in many kinds of meetings, where these applications are used to present data that the group discusses. For example, a design meeting might be called to discuss a prototype created in Visual Basic or Director, and the participants want to take turns trying out the interface, or just pointing to problem areas. Another example is a planning meeting, where a budget might be displayed in Excel and participants modify the numbers relevant to their part of the plan.

These kinds of meetings may be co-located, where everyone is in the same room looking at a projected display, or remote, with each person at their own computer at a different site. As more and more conference rooms come with computers and built-in projectors, it will be increasingly easy to bring real-time computer displays of applications into meetings. Even small meetings in an office around a desktop computer can have a collaborative character and consequent floor control issues. Similar problems arise with technologies that allow remote users to share legacy applications. This is often called "shared screen" or "shared window" collaboration using "view sharing" software, and many different types have been created [4]. A recent example is Microsoft's NetMeeting, which allows remote users to see on their computer an application running on any user's computer. The user running the application can specify how the cursor is controlled among all the users. NetMeeting also provides a separate "shared whiteboard" application, where everyone can draw at once. One piece of evidence that floor control is a challenging issue is that different versions of NetMeeting have apparently used different mechanisms for floor control, presumably due to user feedback about problems with the techniques used in earlier versions.

As part of the Pebbles project, we have been studying the use of Personal-Digital Assistants (PDAs) such as Palm Pilots and Pocket PC devices (formerly called WindowsCE) to control a PC [13]. For example, we have applications where each user has a separate cursor, for custom PC applications that support this. Another application allows each user's PDA to control the PC's real mouse and keyboard, to support legacy applications. Initially, this application provided no floor control, and we observed some problems as a result. Therefore, we investigated what techniques would be appropriate for choosing who is in control.

Submitted for Publication

Floor control has always been an issue for shared-user applications [4]. However, we were surprised to find very few prior studies comparing different floor control mechanisms, and none that compared the many different techniques that are available. We found no guidance on which one would be best in our situation.

Therefore, we classified the floor control mechanisms discussed in the literature, and selected the ones that seemed most relevant to our tasks. We then performed a new study, reported here, that compared eight mechanisms using a collaborative game of solving an on-screen jigsaw puzzle. In summary, the results are that the condition where each person could control their own separate cursor, and work in parallel performed far better than single-cursor conditions, often taking only half the time. Among the floor-control methods, we found no significant differences in speed or user preference. This was surprising since other work [5] suggested that the fastest method would be to have automatic passing of the floor when the cursor was not in use.

RELATED WORK

Many different kinds of floor control have been proposed, but few have been evaluated in studies. Greenberg discusses a number of different floor control mechanisms, but concludes that "surprisingly, there has been no attempt to evaluate these different methods in existing shared view systems" [4]. Boyd [1] classifies floor control mechanisms along a number of dimensions, and introduces "fair dragging," which automatically grabs the floor when dragging starts, and gives pending requests for the floor to users in the order requested. A special form of moderator control was developed to support Mbone videoconferences with hundreds of remote participants [8]. Other papers recommend providing users with multiple floor control mechanisms, since different mechanisms might be appropriate for different kinds of meetings and software [6] [5], but most of the recommendations appear to be based on intuition rather than studies or real data.

Many CSCW applications have multiple cursors, so each user can operate independently. Examples include shared whiteboard systems for drawing (e.g., Tivoli [16] and NetMeeting), and shared text editors such as ShrEdit [15]. An important issue in these systems is locking the content to prevent inconsistent edits. Various CSCW software architectures, such as Timewarp [3], Suite [10] and Prospero [2], provide different mechanisms for such consistency management, and often do not provide any floor control mechanisms. In our study, since are using a single application with co-located users, we only need a simple locking mechanism which prevents multiple users from grabbing the same object. A few software architectures for groupware, such as GroupKit [5], have been designed to support multiple floor control mechanisms, so the developer can choose. Our underlying architecture is the Amulet system enhanced with groupware capabilities.

Recently, there have been a few small studies of floor control mechanisms. Inkpen [7] found that with 9 to 12 year old children, pairs of co-located girls solved more puzzles (from the computer game "The Incredible Machine") when each had their own mouse using a "give floor" mechanism, but boys solved more puzzles using a "take" mechanism. Another study [9] found that for adults collaborating on the classic "survival game" decision-making task, face-to-face interaction (with no CSCW tools) worked best. Of the conditions where the subjects used a textual chat interface for remote collaboration, "give" (where a turn was held until explicitly relinquished) worked best, followed by free-for-all, with "take" (explicit grab) doing least well.

There have been many studies about communication in distributed meetings, where people communicate using various mediums such as video, audio or computer chat. One relevant finding from such studies is that people usually take turns and do not work in parallel [14]. This might suggest that for our application, giving each person their own cursor might not be necessary, but our results show otherwise.

FLOOR CONTROL MECHANISMS

There are a surprisingly large number of different possibilities for floor control. The literature [1] [5] [7] [9] lists many possibilities, but we created a new classification that seems more comprehensive, and which better distinguishes the different choices.

Floor control policies have three primary independent dimensions: how people give up or release control, how people acquire control, and what happens to requests if control is not available. Prior papers have not identified all of these options.

The options for releasing control include:

1. **Explicit Release**: The current floor holder must explicitly release control before anyone else can acquire it. This has also been called "give floor" [7]. Usually, the user pushes a button to signal being finished.
2. **Implicit Release**: The system notices that the user is not busy and releases the control automatically. This might occur after the system detects a pause of activity by the current floor holder. For example, the system might reserve the cursor for one user while that user is moving the mouse, and then wait for a short time-out after movement stops, to make sure the user does not start moving again, before releasing the floor. Alternatively, the cursor might be reserved while the mouse button is held down (while dragging an object) [1].
3. **Explicit Loss**: Whether or not the user is finished, the control can be explicitly removed and given to someone else. For example, a moderator or timeout might determine that the user has had control for too long.

Once the floor control is available, it then is assigned to a user. There are various options for this as well:

1. **Moderator**: A designated participant acts as a chairperson who is responsible for deciding who gets control. This would probably be the best method to

use for classrooms, presentations and other situations where someone must be in charge.

2. **Explicit Request**: A user can request control of the floor using a button or other explicit means.
3. **Implicit Request**: A user indicates an interest in having the floor by performing an input event, such as moving the mouse or typing, and this implicitly signals the desire to have control.
4. **Rule-Based**: The next person to have control is determined by some sort of rule. The most common is "Round Robin," where each user gets a turn in a particular order, even if they do not have anything to contribute just then. Other rules are possible, such as giving people different priorities.

For some combinations of release and request, there are three options of what can happen to the requests:

1. **Immediate Grant**: The request is granted immediately. This only works with the Explicit Loss release policy.
2. **Queued**: The requests are put in a queue, usually in first-come, first-serve order. When the floor is released, then the person at the top of the queue gets control.
3. **Ignored**: If the requests are not queued or granted, then they might be thrown away. Therefore, with this policy, it only works to request the floor when it is available.

By combining these release and request mechanisms, all of the existing floor control policies can be constructed. Some examples from the literature include:

- **Free-floor**: combines Explicit Loss with Implicit Request and Immediate Grant. Any participant can enter input at any time, and control immediately passes to that user. The interaction is mediated by social mechanisms such as eye contact and talking. In human-human communication, social mechanisms are sufficient for smooth flow of most communication, so it might be expected that free-floor would work best, at least for situations when the people can easily see and hear each other. For situations when the participants are remote, and such channels are not available, then prior research [9] suggests that free-floor will not work well, however.
- **Pause detection**: combines Implicit Release with Implicit Request and Ignored. The floor is made available automatically when the user is finished, and the next person to do something gets control. Trying to do something while someone else has the floor is ignored.
- **Preemptive**: combines Explicit Loss with Explicit Request and Immediate Grant. Anyone can grab control of the floor at any time, even while someone else is doing something. This is also called "take floor" [7].
- **Fair Dragging** [1]: combines the Explicit Loss due to a time-out with Implicit Request and Queued. Boyd calls this policy the most fair because it gives each person a turn who wants one [1].

Of course many other combinations are possible. A further variation is to use different policies for different applications, and even different policies for different users. Both of these are exemplified by current version (v3.01) of Microsoft's NetMeeting. For its shared whiteboard, each user can have a separate cursor, and normally everyone can draw at the same time. A menu command allows a user to lock the whiteboard so only that user can draw, and that user must unlock the whiteboard to free the other cursors. This might be considered a form of Explicit Release combined with Explicit Request and Ignored. NetMeeting also allows any legacy application's windows to be shared. Let's call the user whose computer is running that application the "owner." The owner can decide to allow others to control the application, and then another user can get control using a menu command. If a different user wants control, he or she will have to go to the menu and explicitly ask for control and then will get it only if the first user agrees to release control and other pending requests are queued (Explicit Request + Explicit Release + Queued). However, the owner can grab control back at any time (Explicit Loss + Explicit Request + Immediate).

Another option is for a system to let the user choose among multiple mechanisms, as recommended, for example, by [6].

PEBBLES

The Pebbles project[11] is studying the use of one or more hand-held computers simultaneously with a PC. An earlier paper [13] presented our applications that allow a PDA to be used as a remote mouse and keyboard to a computer. We also have applications that allow each user to have a separate cursor, either to scribble on top of the screen, or to control applications that support multiple cursors. The concept is that people will be carrying their PDAs into meetings, and we might be able to use them to control the PC.

Other Pebbles applications are aimed at the individual use of a PDA with a PC [12], for example, while the PDA is in its cradle next to the desktop computer.

For the purposes of the current study, we were interested in the floor control issues arising out of each user controlling the PC from their PDA. We modified the Remote Commander application described earlier [13] to remove features not needed by the study, and to add floor control buttons (see Figure 1).

Figure 1. The Palm running the puzzle control program.

The way that the Palm is used as an input device is as follows: Moving the stylus across the Palm screen causes the cursor on the PC screen to move a corresponding relative amount. This is analogous to the way that the small touchpad works on some laptops, like the Macintosh PowerBook. We put a small piece of tape across the Graffiti input area at the bottom of the Palm screens used in this study, so the users would not go out of the active area when using the Palm while looking at the PC's screen and not looking at the Palm's screen.

Since the Palm screen was used to simulate a touchpad, we needed a separate signal for pressing the mouse button. We provide three different ways to signal pressing a mouse button. First, tapping (pressing and releasing the stylus in the same place) on the blank area of the Palm Pilot screen with the stylus causes a click event. Second, we provided an on-screen button that could be used to signal the button press (see Figure 1), but this required the user to look at the Palm screen. Third, we mapped all of the Palm's physical buttons to mouse press (see Figure 1). Users could hold the Palm in their non-dominant hand, and use their thumb or a finger of that hand to press the buttons, while the dominant hand held the stylus. The most intuitive method to the subjects was tapping, but this was often difficult to use for subjects with no prior Palm Pilot experience. We observed that all the subjects preferred to use the tap method, but about two-thirds eventually decided to use the physical buttons because they had difficulty tapping without moving. None used the on-screen drag button.

EXPERIMENTAL DESIGN

Task

We wanted to find a task for this study that would have the properties we observed in real meetings, where a PC is used to display information, and the people in attendance provide input. Often, the attendees will want to control the mouse and keyboard, either to take turns, or occasionally to work in parallel entering information or adding their individual annotations. Therefore, we wanted a task that had both parallel and sequential components, where the people might want to work at the same time at some points, and other times might want to take turns.

Another aspect of meetings is the trade-off between group goals and individual goals. Since people in most meetings are there to achieve a common purpose, usually for the same company, there is a general goal to cooperate and succeed as a group. On the other hand, often each person wants to make sure that his or her input is heard, or try to prevail in debates. Therefore, we wanted a task that had both cooperative and competitive aspects.

Another important consideration is that we wanted to use a within-subjects design, where each group would see all the conditions. This would decrease the number of subjects we would need. Further, effects of individual differences across conditions would be minimized in within-subjects design.; this would permit comparative analysis to be performed. On the other hand, this meant that we had to find a task that groups could do over and over, once for each type of floor control. Since there are many different floor control options, this also meant the task had to be relatively quick to complete.

Taking all these requirements into account, we eventually settled on the task of doing a jigsaw puzzle. We first did a think-aloud study where a group of four participants attempted to solve various store-bought physical jigsaw puzzles with different numbers of pieces. We observed significant parallel behavior, as different people would work on different parts of the puzzle at the same time. Puzzles have a cooperative aspect in the shared goal of finishing the entire puzzle, and a competitive aspect as people sometimes go for the same piece, or interfere with each other when trying to work on the same part of the puzzle. Furthermore, the puzzle activity where participants try a series of trial and error steps finding where a piece fits might be considered analogous to a design meeting where participants work on different aspects of the product, modifying them by mixing and matching the various elements.

In the think-aloud study, we determined that about 50 pieces would take about 10 minutes, and that people divided the work by regions and the colors of the picture. We also observed that the difficulty varied depending on the picture on the puzzle, and there were significant individual differences in puzzle-solving aptitude.

We next developed a set of computerized jigsaw puzzles of equal difficulty. We decided to use abstract geometric shapes as the pictures, because then we could be more sure that the puzzles had equal difficulty (see Figure 2). Some additional requirements for the pictures were that there should be distinctive regions of the puzzle with different colors, to permit the division of labor among participants where different people would work on different parts. We also wanted to make sure that no two pieces were identical

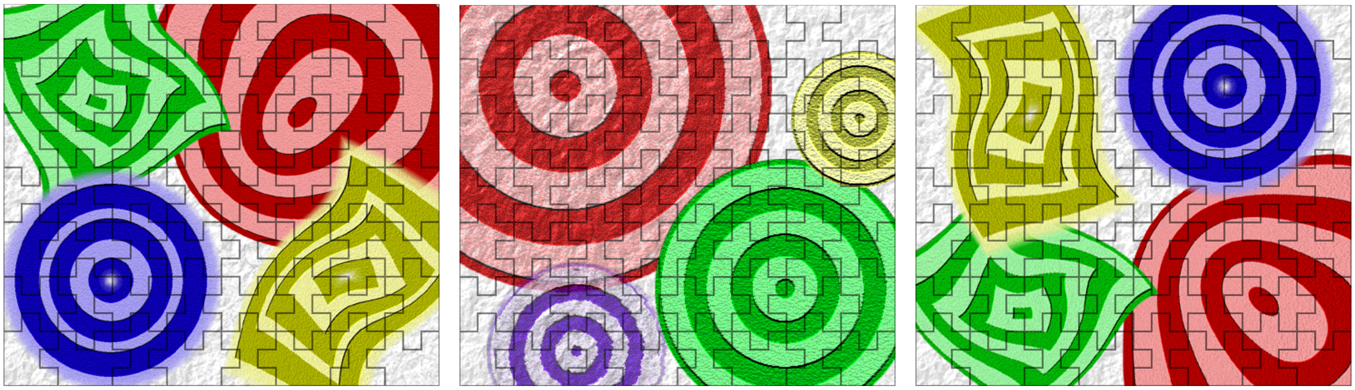


Figure 2. Three of the puzzles used in the study. The others were similar. On the screen, the areas are different colors.

(in particular, that no two pieces were a solid color). This would make sure that it was always apparent from the picture when a piece was in the wrong place.

All of the puzzles use the same shapes for the pieces, where there were basically only 2 puzzle shapes, and the software ensured that only the correct pieces would connect.

In order to give the subjects motivation to *cooperate*, we promised a bonus of $6 each if they finished all the puzzles in less than 6 minutes. This time was based on predicted times for completion of puzzles based on the pilot tests, but in practice, none of the groups was able to achieve this goal. We also awarded the group with the fastest average time a $5 bonus per member. We tried to get participants to *compete* within their group by paying each subject an amount depending on how many pieces that subject had attached. Figure 3 shows the display which shows the subjects with the number of pieces they have attached under their names.

Figure 3. The Puzzle program on the PC in multi-cursor mode with three users. Marsha's cursor at the lower-right is dragging one green piece. YuShan's cursor at the top-center is dragging three red pieces connected together. Brad's cursor at the lower left is dragging one piece. The user-list window at the upper right shows the number of pieces each player has attached, and the elapsed time.

The operation of the puzzle program is as follows. A puzzle is loaded and is displayed as it will look when completed (as in Figure 2). Then the puzzle is shuffled, which displays it like in Figure 3. To move a piece, the cursor is moved over the piece, and the mouse or Palm button is clicked (pressed and released). This "lifts" the piece up and shows the piece with a shadow (see Figure 3). The piece then follows the cursor. When the user wants to drop the piece, the mouse or Palm button is clicked again. If the piece is dropped next to the correct neighboring piece, then the two pieces snap together. Picking up pieces that have been snapped together picks up the whole group, which can be placed down to attach to other pieces as well.

Figure 4. The user window on the PC when using floor control with a single, shared cursor. The user shown in reverse video currently is controlling the cursor. The user number is to the left of the name inside the button, and the number of pieces that person has connected is to the right of the button. The elapsed time is at the bottom.

The puzzle program was implemented using the Amulet user interface toolkit, which has been modified to support multiple input streams. We attached each of the Palm Pilots to the PC using long serial cables. The puzzle program was augmented with multiple cursors, for the conditions where each user would have their own independent cursor (see Figure 3). The cursors had different shapes and colors, and the first few letters of the user's name at the bottom. When there were multiple cursors, a user window (on the upper-right of Figure 3) displayed the user's cursors and the number of pieces they had connected.

To simulate sharing an existing application, where there is not an option to let each person have their own cursor, the puzzle program also supports various floor control modes with a single cursor. In these conditions, the movement on the Palm Pilot causes the real cursor to move on the display. In these cases, the user window changes to show which user has control of the cursor (see Figure 4).

Conditions

Each group did the puzzle for all eight conditions (a within-subjects design). To control for learning and fatigue effects, we varied the order of the conditions for the groups using a Latin square. Also, we varied the particular puzzle picture used with each floor control condition, although we strove to make designs of equivalent difficulty.

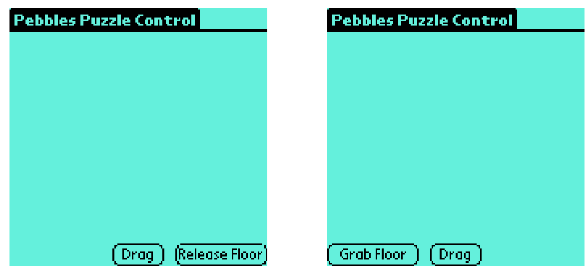
The conditions which used floor control (one shared cursor) were:

1. **Moderated:** (Explicit Loss + Moderator + Immediate Grant) One of the subjects was picked to be the moderator. We chose the moderator by observing which subject generally performed the best in the other conditions. (Since there were fewer groups than conditions, we could arrange for the moderated condition never to be first.) The moderator chose which user had the floor by typing that user's number on the computer's keyboard. The user's number was displayed on the screen next to the user's name (inside the buttons to the left of the names in Figure 4).
2. **Free-floor**: (Explicit Loss + Implicit Request + Immediate Grant) All of the inputs from all of the users were mixed together to control the one cursor. For example, if one user moved diagonally to the upper left while the other user moved diagonally to the upper right, the cursor would move straight up, and the control would alternate between the two users as each user's mouse events were handled. We had tried other ways of combining the movements from multiple people, but this seemed the least surprising.
3. **Pause detection**: (Implicit Release + Implicit Request + Ignored) Whoever had the floor kept it until that person stopped moving for more than one-half second. In addition, if a person had picked up a piece, they would have the floor until they dropped it, even if they stopped moving. After the timeout, the floor would freed (so no-one had the floor), and then the first person to move or press a button would get the floor.
4. **Explicit release**. (Explicit Release + Implicit Request + Ignored) A button was displayed on each Palm Pilot screen which allowed the user to release the floor (see Figure 5a). Once released, as in the Pause-Detection condition, whoever moved or pressed first would grab the floor.
5. **Preemptive**: (Explicit Loss + Explicit Request + Immediate Grant) A button was displayed on each Palm Pilot screen which allowed the user to grab the floor (see Figure 5b). Once grabbed, the user had control until someone else explicitly grabbed the floor away. The floor could be grabbed at any time, even while someone else is dragging a piece.

We decided that the policies using queues, rules and time-slices were too rigid for the kinds of free-flowing meetings we wanted to support with the Pebbles software.

We also added some non-floor control conditions:

6. **Multi-Cursor (parallel)**: All participants have their own cursor, and can move pieces independently.
7. **Multi-Cursor with taking turns**: All participants had their own cursor, but we asked them to nicely take turns anyway and *not* work in parallel. This was not enforced in the software, though.
8. **A physical puzzle**: The subjects did not use a computer for this condition. We took one of our on-screen puzzles, printed it on a color printer, glued it to cardboard, and cut it up. In this condition, the subjects put together the cardboard puzzle on a table.



(a) (b)

Figure 5. The Palm screen when using (a) Explicit Release and (b) Preemptive floor control.

Apparatus

First, the subjects used separate computers to learn how to use the Palm to control the cursor and operate the puzzle. They practiced separately putting together puzzles for about 5 minutes.

Then, they took their Palm's over to a table, on which was a laptop computer connected to a projector. The projector displayed the screen on the wall in front of all the subjects so that all the participants would have access to a shared visualization of the progress of the game. The subjects held the Palm Pilots in their hands or on their laps and not on the table. The subjects were sitting right next to each other, and could talk, look around, and gesture normally. This was to try to best approximate the kinds of co-located meetings which the Pebbles software is designed to support.

Hypotheses

We had a number of hypotheses before we ran the subjects for this study:

- Subjects would perform best with the physical puzzle, since this is the most natural and familiar.
- Subjects would perform next best in the Multi-Cursor parallel condition, since they could all work independently and in parallel. However, if the result that people do not work in parallel as observed in studies of remote

communication [14] transferred to our task, then this condition would be not necessarily be faster.

- The Multi-Cursor with taking turns condition would be next, and would be slightly faster than the floor-control conditions because natural social mechanisms could be used for taking turns without any interference from the technology.
- Of the floor-control conditions, Pause-detection would work best, as predicted by [5]. Because a similar technique mediates turn-taking in human-human conversations, it is expected that this method would be the most natural and intuitive to use.

Subjects

We had 4 groups of 3 people each for this study. The subjects were recruited from the CMU community using various advertisements. We required that the groups be composed of people who were friends with each other, since in most real meetings, the participants know each other. The subjects were paid $7.50 plus a bonus depending on their performance. The typical payment was between $10 and $20 per person. Of the 12 subjects, 3 were women and 9 were men. Two of the groups were mixed gender and the other two were all men. The median age was 23. Very few had used Palm Pilots or other PDAs before (the average self-rated "proficiency" with PDAs was 1.25 on a 0 to 9 scale where 9 was very proficient). However, almost all of the subjects rated themselves very proficient with computers (average 7.2).

Method

The subjects first filled out a pre-questionnaire that acquired demographic information including their level of experience with the Palm Pilot and their general computer skills. Next, the subjects trained independently using separate computers about how to use the Palm Pilot to control the cursor. Then, the subjects moved to a table, and performed the experimental task with all eight conditions. (One group had to omit doing the physical puzzle.)

Subjects were told to complete each puzzle as quickly as possible. In order to motivate each person to actively participate in the group session, a monetary bonus of 3 cents was earned for every piece that that person placed correctly in the puzzle. This was to prevent the same individual(s) from dominating the entire puzzle session. At the same time, group cooperation is encouraged by rewarding the group $6 if all puzzles were completed under 6 minutes each.

At the end of the study, participants were asked to rate their satisfaction, as well as the efficiency with which they completed the study, using each of the mechanisms.

RESULTS

Figure 6 shows the times for each of the groups to finish the four conditions. Figure 7 plots the average time in minutes over all the groups. The multi-cursor-parallel condition is faster than each of the others and this is statistically significant using a paired 2-tailed t-test at significance level $p<.05$. For example, the closest time to multi-cursor-parallel is pause detection. There was a highly significant different between multi-cursor parallel (M=4:07) and pause-detection (M=7:31) ($t=-16.713$; $p<.001$). However, the times for the other conditions were not statistically different from each other.

Figure 8 shows a plot of the 12 subjects' rating of each of the 7 computer methods. On the final questionnaire, subjects were asked to rate how efficient they thought each method was for them, and how satisfied they were with the method, using a 9 point Lickert scale, where 9 was the best and 1 was lowest. There was a high correlation between the subjects' efficiency ratings and their satisfaction ratings for all the conditions ($R=.80$; $p<.01$). Among the ratings, a paired t-test showed that the multi-cursor parallel condition was rated significantly higher on efficiency and satisfaction than all the others ($p<.05$). The lowest-rated condition, explicit release, was not statistically different from the two with nearest ratings, pause-detection and multi-cursor-turns, but was significantly lower than the other four at the .05 level.

Finally, we asked the subjects to rate the 7 methods from their most favorite (1) to the least favorite (7). Figure 9 shows the average of the subjects answers for each method. The only difference that is statistically significant in the largest and the smallest. A paired t-test showed that multi-cursor parallel was significantly better liked than pause detection ($t=-2.455$; $p<.05$).

DISCUSSION

Hypotheses

Of our hypotheses, only one was supported by the study.

We conjectured that the physical puzzle would be the fastest, but the physical puzzle second fastest for one group, and was the *slowest* puzzle for another group. The physical puzzle was actually a harder task than the on-screen puzzles, because the pieces could be rotated on the physical puzzle, but on the on-screen puzzle, they were always in the right orientation. Furthermore, since the pieces didn't lock together like a normal jigsaw puzzle, many groups had trouble with the puzzle falling apart as new pieces were added. We therefore feel that it is not fair to compare the on-screen and physical puzzle times.

Our hypothesis that the Multi-Cursor parallel condition would be the fastest computer condition was strongly supported. The time for each group on this condition was the smallest, and often half the time of the other conditions. This confirms the findings in prior literature that recommends giving each person their own cursor when possible. Unlike other multi-person tasks such as communication [14], the puzzle task seemed most natural to do in parallel.

Free-floor	Grab	Explicit Release	Moderator	Pause-Detection	Multi-Cursor-Parallel	Multi-Cursor-Turns	Physical Puzzle
8:20	8:36	6:40	7:59	8:41	5:09	6:24	4:40
7:30	6:39	10:27	8:04	7:01	3:11	8:26	14:00
7:41	10:14	8:49	8:03	7:13	4:21	7:56	*Omitted*
6:38	6:39	9:54	6:57	7:09	3:49	7:59	6:22
7:32	**8:02**	**8:57**	**7:45**	**7:31**	**4:07**	**7:41**	**8:20**

Figure 6. Times (min:sec) for each group to finish the task in each condition. The bottom row is the average.

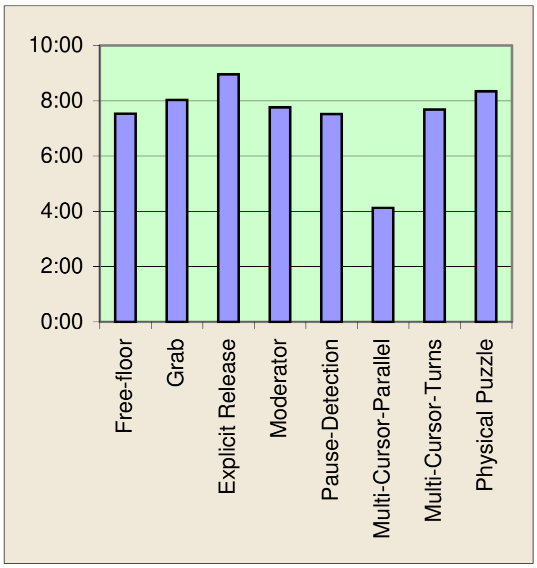


Figure 7. Average time (min:sec) for the groups to complete the task for each of the conditions. Shorter bars are better.

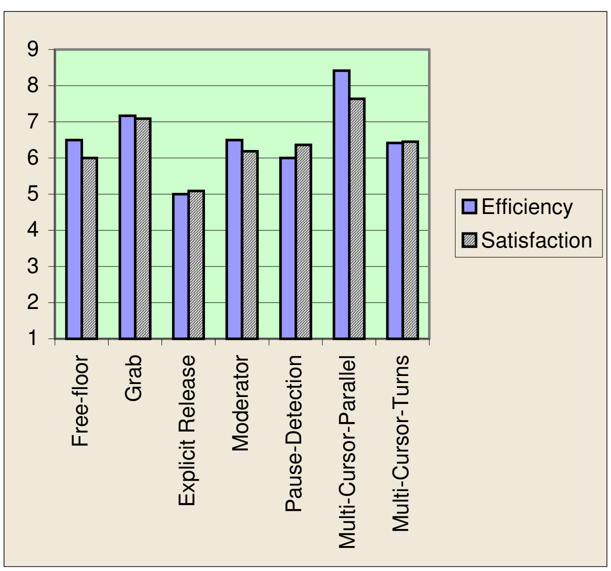


Figure 8. A plot of the subjects rating of the efficiency and their satisfaction with each method. Longer bars are better.

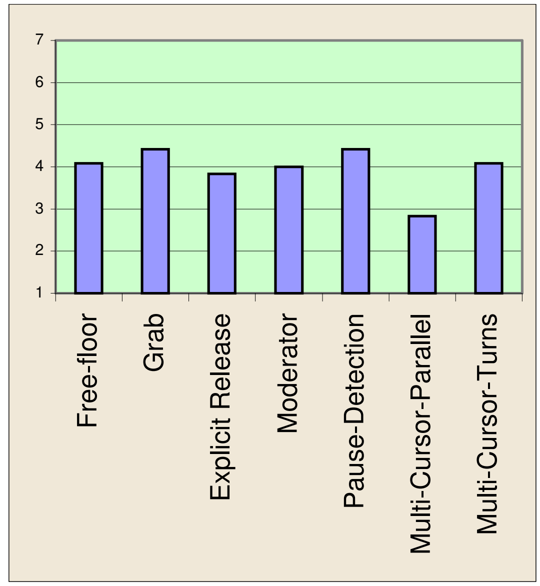


Figure 9. Each subject was asked to rank order the different methods. This plot shows the average. 1 = best liked, and 7 is least liked, so shorter bars are better.

The Multi-Cursor with taking turns condition was not very successful, and our conjecture that this would be one of the fastest was not supported. We thought that this might be like regular turn taking with separate physical pointers (like each person's finger) so it would be a natural mechanism, but in fact, the turn-taking often broke down, and the subjects wasted time in deciding whose turn it was. Some subjects complained that since they each had a cursor, they should have been allowed to use it.

The Pause-detection condition had the lowest average score, so the trend is in the direction we predicted, but there is no statistically significant difference between it and the other single-cursor floor control conditions. We did not see any evidence, as might be expected from suggestions in prior work [5] that this method was more natural. We observed a number of problems with our implementation of this method, which might be fixed in a future implementation. There seemed to have been insufficient feedback about when the user was done (the only feedback was that the user's name turned from black to white in the on-screen menu of Figure 4), so others did not know when to start. Also, the ½ second pause after movement stopped appeared to be unnatural and annoying. Furthermore, we noticed that some people would absent-mindedly have their stylus moving on the Palm even when they didn't want to do anything, so the system would think they should have the floor. Another behavior we observed was one subject intentionally moving the stylus when it was not his turn, as a way to grab the cursor as soon as it became available.

General Observations

We have a number of general observations about how the subjects approached this task.

There was a great deal of discussion and communication among the subjects about the task and about turn-taking. We saw many instances, across most of the conditions, where one person would verbally direct another about where a piece would go, rather than, for instance, taking the piece and putting it in. Even though they could use the cursor to point, users often still pointed their finger at the screen, which was much less accurate but much faster. There were only a few occasions, mostly during the multi-cursor with turn-taking condition, where one person would use the cursor to show another person what to do.

When we asked why they were directing rather than taking over the cursor themselves, they said that it was easier to tell the other person what to do rather than taking the cursor over since that could break the continuity. Also, subjects said they could see better when they were not doing the puzzle themselves.

To control the turn-taking, they often used phrases like "wait, I see something" and when done, they would say "ok, go" to augment the on-screen feedback that the cursor was available. For example, in the explicit-release case, participants with control would ask if anyone wanted control (e.g., one subject asked "who wants to take over").

This alleviates the chaos that might have arisen for the implicit strategies like free-floor. We only noticed a few instances of surprise and contention where multiple people interfered with each other. We conjecture that we would see quite different results, and possibly significantly more difficulty, if we tried these methods when users were not co-located and had more limited communication bandwidth.

As we hoped, the groups did discuss strategies of how they would divide up the labor and either assigned corners of the puzzle or different colors to the different people. This occurs repeatedly across all groups and conditions. At the beginning of each trial, each member in the group would assign themselves a distinct portion of the puzzle. As the game progressed however, while some groups stick to their distribution of puzzle parts, others changed them whenever someone else saw a piece of the solution that would help accelerate the completion of the puzzle.

Although we tried to set up a task that would let all the subjects be equal peers, in every group a leader emerged. This is consistent with prior research on group dynamics. The leader was implicitly established early as the person who got the most points in the early trials, and usually the person continued to dominate throughout. Thus, in our study, the leader emerged through talent with the task rather than due to having a forceful personality.

Subject	Free-floor	Grab	Explicit Release	Moderator	Pause-Detection	Multi-Cursor Parallel	Multi-Cursor-Turns
A		26					
B	20						
C			52				
D	27						
E		23	23				
F							22
G				35			
H	8						8
I					32		
J	21						
K							19
L			35				

Figure 10. Comparison of each subject's best mechanism (shown by their score of the number of pieces they connected) compared with the mechanism they rated best (shown by the shaded interior of the square). Each group's fastest time was the Multi-Cursor-Parallel. The mechanism with which the group did second-best is shown by the multiple borders around the squares.

We chose this leader to be our moderator, since we assumed that in a real meeting, the moderator was likely to be the leader of the meeting. Usually the leaders' scores were lower when they were the moderator, suggesting that they might have felt it was unfair for them to take all the pieces when they were in charge. In some instances, the moderator decided who would get the cursor since none of the participants requested the control. In other groups, the others did ask for control and the moderator gave control to whoever asked.

Inspired by Inkpen's study [7], we were interested in seeing if there were any gender differences. Unfortunately, we were unable to get an equal representation of women. Of the two mixed-gender groups, once a women emerged as the leader, and once a man did. We also found no differences in performance or preference between men and women on different floor control methods, suggesting that adult men and women might not differ as much as Inkpen found that children did.

We specifically recruited groups of people who were friends. In one pilot study using a group of 3 strangers, the group dynamics were quite different. Instances of turn-taking behavior were infrequent, and therefore the session was more likely to be dominated by the same individual. This might be due to the fact that among strangers, people are more awkward, and therefore more hesitant, in requesting the control of the cursor. Further, it is less likely to have a group of strangers convene during a meeting. Therefore, using 3 friends seemed more representative of real meetings.

We observed people having more trouble in the first trial with controlling the cursors than in later trials, but this effect was not statistically significant. Most users were observed to become fairly proficient during the first or second trial. Most of the subjects in the study did not have experience with using Palm Pilots or other PDAs. In our envisioned real meeting, where people would bring in their own PDA that they are familiar with using, we would expect better performance across all kinds of floor control.

It is interesting that people's preferences for which mechanism they liked best did not necessarily correlate with which method they got the most pieces, or with the fastest method. Figure 10 shows the comparison of which condition each subject did best in, compared to which one the subjects rated highest, and there is no overlap. All groups did best in the multi-cursor-parallel condition, which 7 out of 12 rated best. One additional subject did pick the method with which they did second best. Therefore, people's preferences are not necessarily a good predictor of the individual's or group's performance.

CONCLUSIONS

In conclusion, for a task that can be performed in parallel by a group, we showed that the most effective strategy is to let everyone work in parallel with their own cursor. When required to take turns sharing the cursor, no particular floor control strategy stood out as superior to the others. Furthermore, users had different preferences and opinions about which ones work best, although often people's preferences did not match with performance. This suggests that users might be happiest with a choice of mechanisms, but that they might not choose in practice the best mechanism for the job. This study focused on a task with naturally parallel elements for a small number of co-located users who were peers. Future studies should be performed on floor control mechanisms for groups with different parameters.

ACKNOWLEDGMENTS

For help with this paper, we would like to thank Bonka Boneva, Rob Miller, Franklin Chen, Ken Koedinger, Bob Kraut, and Al Corbett.

REFERENCES

1. Boyd, J. "Floor control policies in multi-user applications," in *INTERACT '93 and CHI '93 conference companion on Human factors in computing systems.* 1993. Amsterdam, The Netherlands: pp. 107 - 108.

2. Dourish, P., "Using Metalevel Techniques in a Flexible Toolkit for CSCW Applications." *ACM Transactions on Computer-Human Interaction*, 1998. **5**(2): pp. 109–155. June.

3. Edwards, W.K. "Flexible Conflict Detection and Management In Collaborative Applications," in *Proceedings UIST'97: ACM SIGGRAPH Symposium on User Interface Software and Technology.* 1997. Banff, Alberta, Canada: pp. 139-148.

4. Greenberg, S. "Sharing views and interactions with single-user applications," in *Proceedings of the ACM/IEEE Conference on Office Information Systems.* 1990. Cambridge, MA: pp. 227-237.

5. Greenberg, S. "Personalizable groupware: Accommodating individual roles and group differences," in *Proceedings of the ECSCW '91 European Conference of Computer Supported Cooperative Work.* 1991. Amsterdam: Kluwer Academic Press. pp. 17-32.

6. Handley, M., Wakeman, I., and Crowcroft, J. "The conference control channel protocol (CCCP): a scalable base for building conference control applications," in *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication.* 1995. Cambridge, MA: pp. 275-287.

7. Inkpen, K., *et al.* "Turn-Taking Protocols for Mouse-Driven Collaborative Environments," in *Proceedings of Graphics Interface '97.* 1997. Kelowna, BC, Canada: pp. 138-145.

8. Malpani, R. and Rowe, L.A. "Floor control for large-scale MBone seminars," in *Proceedings of the conference on Multimedia '97.* 1997. Seattle, WA: pp. 155 - 163.

9. McKinlay, A., *et al.* "A Study of Turn-taking in a Computer-Supported Group Task," in *People and Computers VIII, Proceedings of the HCI'93 Conference.* 1993. Cambridge University Press. pp. 383-394.

10. Munson, J. and Dewan, P. "A Concurrency Control Framework for Collaborative Systems," in *Proceedings CSCW'96.* 1996. Boston, MA: pp. 278-287.

11. Myers, B.A., *et al.*, "Using Hand-Held Devices and PCs Together." *ACM Communications of the ACM*, 2001. : pp. To appear.

12. Myers, B.A., *et al.* "Extending the Windows Desktop Interface With Connected Handheld Computers," in *4th USENIX Windows Systems Symposium.* 2000. Seattle, WA: pp. To appear.

13. Myers, B.A., Stiel, H., and Gargiulo, R. "Collaboration Using Multiple PDAs Connected to a PC," in *Proceedings CSCW'98: ACM Conference on Computer-Supported Cooperative Work.* 1998. Seattle, WA: pp. 285-294.

14. O'Connail, B. and Whittaker, S., "Characterizing, predicting and measuring video mediated communication: A conversational approach," in *Video Mediated Communication*, K.E. Finn, A.J. Sellen, and S.B. WIlbur, Editors. 1997, Lawrence Erlbaum Associates. Mahwah, NJ. pp. 107-132.

15. Olson, J.S., *et al.*, "Groupwork Close Up: A Comparison of the Group Design Process With and Without a Simple Group Editor." *ACM Transactions on Information Systems*, 1993. **11**(4): pp. 321-348. October.

16. Pederson, E., *et al.* "Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings," in *Proceedings INTERCHI'93: Human Factors in Computing Systems.* 1993. Amsterdam, The Netherlands: pp. 391-398.