Single Display Groupware: A Model for Co-present Collaboration

Jason Stewart
Computer Science Dept.
University of New Mexico
Albuquerque, NM 87106
jasons@cs.unm.edu

Benjamin B. Bederson
Computer Science Dept. / UMIACS
Human-Computer Interaction Lab
University of Maryland
College Park, MD 20742
+1 301 405 2764
bederson@cs.umd.edu

Allison Druin
College of Education / UMIACS
Human-Computer Interaction Lab
University of Maryland
College Park, MD 20742
+1 301 405 7406
allisond@umiacs.umd.edu

ABSTRACT
We introduce a model for supporting collaborative work between people that are physically close to each other. We call this model Single Display Groupware (SDG). In this paper, we describe this model, comparing it to more traditional remote collaboration. We describe the requirements that SDG places on computer technology, and our understanding of the benefits and costs of SDG systems. Finally, we describe a prototype SDG system that we built and the results of a usability test we ran with 60 elementary school children.

Keywords
CSCW, Single Display Groupware, children, educational applications, input devices, Pad++, KidPad.

INTRODUCTION

In the early 1970's, researchers at Xerox PARC created an atmosphere in which they lived and worked with technology of the future. When the world's first personal computer, the Alto, was invented, it had only a single keyboard and mouse. This fundamental design legacy has carried through to nearly all modern computer systems. Although networks have allowed people to collaborate at a distance, the primary assumption still remains that only a single individual would need to access the display at any time.

Is this a valid assumption? Do we really work in isolation, without the desire to interact with one another around a computer? When designing technology for elementary school children, we frequently observed two, three, and four children crowded around a computer screen each trying to interact with the computer application [7]. It also appeared to us that children enjoyed their experiences with the computer more if they had control of the mouse and were actively controlling the application. Since there are times when multiple people would like to each interact with a computer application, how does the lack of technological support affect people's collaborative behavior? Could we as technology designers improve collaboration by explicitly designing computer support for collaboration at a single computer display? We believe that we can, and in this paper, we introduce a model for doing so.

We define Single Display Groupware (SDG) to be computer programs that enable co-present users to collaborate via a shared computer with a single shared display and simultaneous use of multiple input devices.

Recent work including our own has begun to explore SDG. In this paper, we attempt to create a framework that ties together these different approaches, and motivate future system designers to include low-level support for SDG.

Scenarios

Let us imagine ourselves in the computing environment of the not-so-distant future where there is universal support for co-present collaboration:

> *At work*, you are visiting the office of a co-worker to get feedback on your latest project. Since the Personal Data Assistant (PDA) you carry uses wireless networking technology, you can easily communicate with your co-worker's computer. After she approves your log-on request, you start up your demo on her monitor, and use the touch screen of the PDA to control a cursor on her workstation. While she uses her workstation's mouse to use your program, you gesture

To appear in CHI 99

with your cursor indicating the areas you had questions about. As you expected, she finds a number of bugs in your code. But since you are both able to interact with the software, you work around the bugs without interrupting her or taking the input device out of her hand.

At the designer's office, you review the plans for the renovation of your living room. After going over some of the paper sketches, the designer offers to show you the 3D model of the renovation on his computer. He thinks it will give you a better idea of how his plans fit in with the rest of the house. As he guides the program into the living room, he encourages you to pick up the extra mouse and investigate the layout yourself. You have some trouble navigating with the unfamiliar software at first, but the designer demonstrates the navigation tools with his mouse and you quickly learn to mimic him. Together you both relocate furniture and experiment with different room layouts and color schemes.

At school, your daughter is finishing work on her latest geometry project. She's having difficulty with the Pythagorean theorem and asks the teacher for help. The teacher is busy helping a group of students working at the other collaborative learning station, so your daughter's friend comes over to help. Her friend picks up one of the unused mice and together they explore the problem. They work together moving around the squares and triangles and measuring the results until they both feel more comfortable with the Pythagorean theorem.

Despite the fact that Computer-Supported Cooperative Work (CSCW) is a thriving field, and networked computing is one of the biggest selling points of computers today, the scenarios described above are not part of today's world of computing. What is missing is that the forms of collaboration we suggest here are *co-present* collaboration. Most research in CSCW today focus on supporting people that are working apart from each other. Computers and networks are very well suited to supporting remote collaboration, but supporting people that are working together requires solutions to new problems.

Based on the computer paradigm discussed in this paper, Single Display Groupware (SDG), we suggest an increase in effort that investigates technology that brings people together and enhances the interaction of people working together.

Related Work

Several projects support people collaborating in the same room. The CoLab project, like other electronic meeting rooms, provided each member with a desktop computer which allowed private work as well as control of a shared display at the front of the room [17]. Earlier shared rooms were built by Krueger as installation art pieces [13]. One drawback of electronic collaborative rooms is that they require expensive, specialized hardware that is prohibitive to many people who could benefit from enhanced support for co-present collaboration, for example school children.

The Liveboard digital whiteboard and the Tivoli application enabled multiple simultaneous users (both co-present and remote) to interact with the shared digital whiteboard [17]. The authors point out that simultaneous use of the whiteboard rarely occurred and they speculated that the lack of adequate software level support for co-present collaboration (of the kind presented in this paper) may have been the cause.

The Pebbles project [15], investigates the use of handheld Personal Digital Assistants (PDAs) as portable input devices in an SDG setting. They have also explored the limitations of current GUI application toolkits and what is needed to make toolkits support SDG.

Another implementation of SDG was MMM [3]. It enabled multiple co-present users to interact with multiple editors on the same computer display by providing each user with an independent input device. The system was never made available to the research community, and no user studies were conducted to investigate the limitations of the idea. MMM was not pursued, but some of the researchers working on it transferred this technology to study the use of multi-handed input for single users.

Other researchers have investigated how SDG technology could influence groups in a learning environment. Work by Inkpen showed that by providing each user with a separate input device gave significant learning improvements, even when only one device could be active at a time. The active device could be toggled through a predetermined access protocol [10]. This is an important result because it indicates that SDG could benefit tasks in which both users are not expected to work simultaneously, such as editing a paper.

Bricker built software architectures that enable building SDG applications that teach collaborative skills [4]. The guiding metaphor of applications built with her SDG architecture is the 3-legged race: the goal is not only to run the race faster, but also to require participants to learn to cooperate in order to be able to run at all. Example applications include a color-matcher in which 3 users must find the RGB values for a given color, and a chord matcher where users find the notes for a given chord.

Rekimoto's multi-device approach enables users to create work on a palmtop computer and then move the data onto a shared public computer, such as a digital whiteboard, using the Pick and Drop protocol [18]. Work by Greenberg and Boyle has also been investigating the boundaries between public and private work by designing applications that can be used collaboratively in both an SDG setting using PDAs or over a network using a workstation [8].

McGrath and Hollingshead conducted a critical review of empirical studies to date of how technology impacted group interaction [14]. An important contribution of this work was a comprehensive listing of variables that should be evaluated when testing the effect that any technology has on group processes. For example, McGrath lists three outcome variables that can be measured: task performance, user reactions, and member relations. The first two outcome variables are commonly measured, but the third, how using technology influences how group members feel about one another, was not commonly measured.

There are other examples of technological support for co-present collaboration that we place in the category of hardware interfaces. Included in this category are vehicles with multiple steering mechanisms, such as aircraft flight controls and driver education cars. Although these systems enable simultaneous co-present users through multiple input devices, they have little or no software interfaces and can teach us little about the design of more general-purpose SDG systems. Other examples of SDG systems include some multiplayer video games. While these are software based, they primarily support users navigating through scenes and shooting things, playing ball, or fighting. They do not support shared creation of information. And, aside from spatial navigation, they do not support much information retrieval. So, while the social issues of video games are interesting to SDG designers, they do not offer us much guidance for interface development.

Input Channels and Output Channels

To better understand the implications that SDG will have on computer system design we need to investigate how SDG applications differ from other applications. User Interfaces consist of *input channels* – which enable users to communicate with the computer, and *output channels* – which enable the computer to communicate with its users.

We define an input channel to be an input device that provides *independent* input to the computer. For example, in current computer systems the mouse and the keyboard would not be considered separate input channels since the keyboard input is dependent upon the mouse for setting keyboard focus. Future computer systems may support an independent mouse and keyboard but current ones do not, so the *typical* current system will be described as having only a single input channel. In some cases, such as laptop computers, a computer has multiple pointing devices, *i.e.,* an external mouse and a trackpad. These devices are also dependent and share the same input channel—either both share control of the system cursor, or only one can be active at a time. This definition covers the observation that dividing up tasks by giving one user the mouse and another the keyboard is not likely to result in a good collaborative experience [16, p. 89].

We define an output channel as a part of the computer interface that uses an independent modality to provide user feedback. Examples would be a display for visual feedback, speakers for audio feedback, and a force-feedback joystick for haptic feedback. Most current computers have the potential of using both visual and audio feedback, but most UIs use little or no audio feedback and rely almost exclusively on visual feedback. There are exceptions to this, such as audio systems for blind users, but these are in the overwhelming minority of existing systems. This could change with future systems, but the *typical* current system will be described as providing a single output channel.

Single Display Groupware and Traditional Groupware

Most computer applications written today are single user applications – they have no special support for multiple users. In contrast, groupware applications are *group aware*, they have a fundamental knowledge of multiple users. SDG is a subset of groupware that focuses on co-present collaboration: multiple users at the same time and place.

Traditional groupware systems create applications that are intended to be run on multiple workstations and can communicate with one another across a computer network. They either communicate in a distributed fashion where each database is synchronized, or with a single centralized server. Similar to a single user application (Figure 1), a traditional groupware application provides both a single input channel and a single output channel for each user (Figure 2).

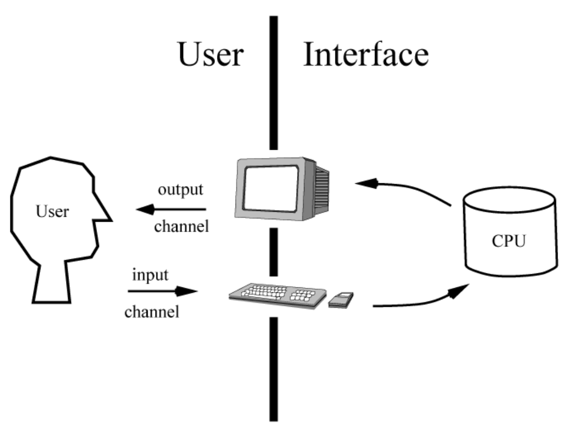

Figure 1: The User Interface for Single User Applications

In contrast, SDG applications provide an input channel for each user through the use of a separate input device, but each must share the single output channel (see Figure 3). These are the qualities that give SDG applications their unique character: the combination of multiple independent input channels together with a single shared output channel. There have been traditional groupware systems which chose to use a shared user interface, or coupled navigation, but the conclusions were that doing so limited the functionality of the application for no apparent gain when the users were *remote* [6, 19].

The Model-View-Controller (MVC) language of the Smalltalk community provides another way of expressing this concept. The Model corresponds to the underlying information of the program, the data. The View corresponds to that part which controls the output channels of the system, while the Controller corresponds to the part that handles the input. Traditional groupware systems have a single shared Model, and since each user has a separate computer, each has a separate View-Controller pair that communicates with the shared Model. SDG systems also have a single shared Model, but differ from traditional groupware systems by only having a single shared View through which the computer must give feedback to all users, and a single shared Controller through which all users interact with the computer. SDG applications could have multiple controllers if an application wanted to replicate all user interface elements and provide every user with a unique copy. This solution seems unlikely to scale as it would quickly take up all available screen space for the user interface.

Figure 2: Traditional groupware

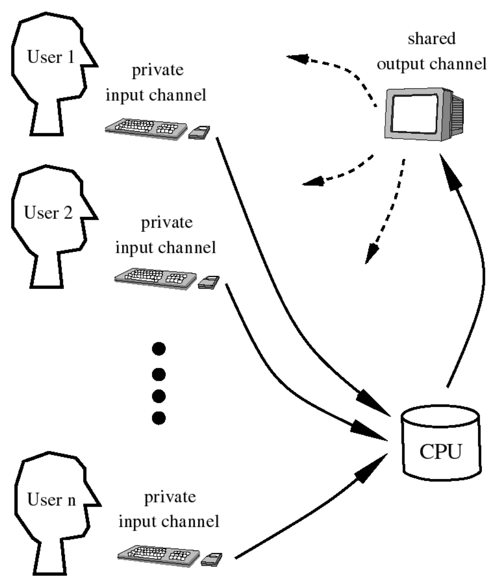

Figure 3: Single Display Groupware

Both the MVC model and the previous discussion about input channels and output channels, help bring out some of the central differences between designing SDG and traditional groupware systems which are:

Shared User Interface. Even though users have separate input devices, the user interface elements that are used to communicate with the computer (menus, palettes, buttons, etc.) must be designed to handle multiple simultaneous users. This restriction corresponds to the single shared Controller in the MVC description.

Shared Feedback. The user interface elements used by the computer to communication state information to users (buttons, palettes, etc.) will likewise be shared by all users and must be capable of relaying information to all users simultaneously. This is a consequence of the shared View from the MVC discussion.

Coupled Navigation. Whenever one user navigates to a different part of the Model the other users will be affected. If the coupling is tight, then all users will navigate together when one navigates. If the coupling is loose, then other users may have part of their Views obscured by one user navigating to a different area of the Model.

Why Single Display?

We could have chosen to expand the scope of this model to include multiple output devices, and called it Co-Present Groupware (CPG). The goal of this work, however, was to study the architectural concerns that arise while supporting multi-user collaboration around a single Personal Computer (PC). The overwhelming majority of current PC systems use a display as the main output channel by which to communicate with users. Some feedback is given using an audio channel, but almost never are touch, taste, or smell used [5, 11]. When users collaborate around a single computer, they consider themselves to be collaborating around the display and not the CPU, hard drive, or CD-ROM drive. For these reasons we chose the single shared display as the central metaphor for this new paradigm.

The single display metaphor is intended to connote several properties of applications that are designed for co-present use. Not only do such groupware systems have shared data, they also possess a shared UI and shared or coupled navigation. What constitutes a single display? If a single computer has multiple displays, does that mean it is not using SDG? What about full wall projection devices that use three projectors to create a single *seamless* display? What constitutes a display? A blind person may use a computer whose only feedback is sound, is SDG therefore not for blind people?

Co-Present Groupware is a more general form of SDG, but since the majority of computers rely almost solely on a visual display for output, we decided that what we lost in generality, we gained back in concreteness. Therefore, we will not include examples which relax the strict conditions imposed by having multiple co-present users at a single display. For example, by using a two-monitor computer each user could be given their own UI, and the shared user interface restriction no longer applies. However, if the use of the second monitor is solely to provide extra physical screen space and not to provide an independent UI, then the conditions still apply and the system could still be considered SDG.

TRADEOFFS IN SINGLE DISPLAY GROUPWARE

Current computer systems do little to encourage collaboration of multiple users. Single user systems provide only one explicit input channel for all users, so if multiple users attempt to collaborate using such a system it is up to the users to develop a sharing mechanism for utilizing that channel. In contrast, SDG applications will have an inherent notion of multiple co-present users and will provide each user with an equivalent input channel. This could have an impact on many aspects of using computers together. Some possible benefits are:

- Enabling collaboration that was previously inhibited by social barriers. For example, in many cultures there is often a reluctance to invade the personal space of another person. The personal space surrounding close friends is smaller than that surrounding co-workers and acquaintances, and the space surrounding strangers is the largest of the three [9, Chapter X]. Due to these proximate effects, many people may be inhibited from attempting to share a computer when another person is sitting in front of it. By explicitly providing for a separate input channel, the personal space around the person may be decreased enough to allow another person to comfortably interact with the computer at the same time.
- Enabling types of interaction that require multiple users. Bricker has explored a number of collaborative interactions that require multiple simultaneous users at a single computer. The goal of her research was to create tools that would strengthen collaborative learning [4].
- Enriching existing collaboration at a computer. For example, turn taking is often viewed as unnecessary and cumbersome [19]. Enabling multiple input devices will in some cases enable work to be done in parallel, making the collaboration both more efficient and more enjoyable in the eyes of the users [7, 22]. Also, a number of studies have indicated the benefit of shoulder-to-shoulder collaboration due to the collaborators enhanced verbal and nonverbal communications [9, pp. 108–111, 20].
- Reducing or eliminating conflict when multiple users attempt to interact with a single application. Often it is difficult to create an appropriate sharing mechanism for the shared channels, or it is difficult to obey the mechanism created [22]. By providing separate channels, potential conflicts are pushed one step further away.
- Encouraging peer-learning and peer-teaching. When existing single user technology is used in a collaborative learning setting, the competition between users to interact with the application can inhibit the learning benefits of collaboration [22]. By providing applications with multiple communication channels, it is possible to enrich learning by diminishing competition for access to the input channels [16, p. 89].
- Strengthening communication skills. Because strong willed users can no longer monopolize a task by merely controlling the input device, users may *have* to communicate more with each other to resolve conflicts.

Along with the potential benefits of the new computer paradigm comes the potential for negative effects:

- New conflicts and frustrations may arise between users when they attempt simultaneous incompatible actions. Working in parallel can be an advantage, but it could also be a disadvantage if each user has conflicting agendas. One serious concern in this area is navigation. Since there is only a single shared output channel (the display), if one user decides to navigate elsewhere in the data space, it may negatively affect the other users.

- SDG applications must squeeze functionality into a very limited screen space, which may result in reduced functionality compared with similar single-user programs.
- Due to increased processing requirements, SDG applications might be slower than a single user version, or a traditional groupware version.
- Because successful SDG implementation depends on low-level operating system and windowing system issues, applications may not be very portable and might exist for only the most popular OSs.
- Completing tasks might take more time, because it is no longer possible for a strong willed user to direct the collaboration by controlling the input device.
- Users may actually collaborate less. Because they can do work in parallel, they may set about completing their own tasks and never communicate with the other users.

In order to build successful SDG applications, these tradeoffs will have to be carefully balanced for each application.

POTENTIAL APPLICATION DOMAINS

SDG is presented to complement the existing single user paradigms, not to replace them. Even so, it is anticipated that there will be collaborative situations in which co-present interaction at a single display will not be as useful as networked synchronous collaboration or asynchronous collaboration. We expect SDG to be potentially useful in at least the following domains:

Creative Domain where users are involved in a creative, expressive, or constructive task such as writing, drawing, artistic expression, programming, and brainstorming. Creative projects often benefit from group activity and input, but the restrictive nature of current systems can limit expression. The potential benefits of using SDG in this domain include being able to work more effectively by working in parallel and eliminate unnecessary turn taking.

Learning Domain where users are involved in the exploration of new material such as a problem solving environment, learning new technology, debugging, or simulations. Learning has been shown to be a domain in which group activity is important [12]. Learning around current computer systems can create an inequality in the partners due to the difference in their skills and the restrictions of only having a single input device [7, 10, 22]. Potential benefits of using SDG in this domain include more effective learning by being able to work at the same time with the same objects, reducing the cognitive difference between partners by giving each parallel access.

Instruction Domain where one user is more experienced than the other and has skill or knowledge to impart such as training to use software, peer teaching in a classroom, or informal help from an instructor.

Sales Domain where a sales person and customer could configure items together. The crucial point is that by allowing customers to play an active role in the selection and configuration process they may be more inclined to choose one product supplier over another.

One area not likely to benefit from SDG is any application that can be best accomplished using a divide-and-conquer approach, such as data entry. Because Inkpen's work shows that adding multiple input devices can benefit tasks in which users are not expected to work simultaneously, there are many areas in which it is unknown how effective SDG could be. For example, when collaboratively editing a paper for a conference, would it be effective to use multiple keyboards, or would it be better to provide the second user with highlighting and gesturing tools instead.

LOCAL TOOLS

In order to investigate how SDG affects group interactions at a single display, we implemented a general-purpose architecture to build SDG applications, the Local Tools architecture. The underlying operating system and event model was built using the Linux operating system, and the X Window System. The application layer was built using the Tk toolkit and Pad++ [2], and was written in Perl. This section will describe some of the important high level issues. While a complete description of the architecture and its implementation are provided elsewhere [21], the following discussion is intended as a motivation for why future systems should include low-level support for SDG.

The current Windows, Icons, Menus, and Pointers (WIMP) metaphor for building GUI applications has a number of limitations when used to build SDG applications. Many toolkits have implemented widgets that do not work in the SDG paradigm (e.g., they use global variables that assume single users). Many applications store user state information in global variables leading to shared interface state such as a single pen color or font. Even if these applications were to store state per user, many feedback techniques are insufficient for displaying multi-user state (e.g., applications would have to be redesigned to have an entire interface per user).

At a high level, the interaction semantics of many widgets are designed for single users. Should locking mechanism be used to prevent multiple users from interacting with the same menubar or scrollbar? Should modal dialog boxes apply to all users or only the user who activated the dialog? What happens when one user interacts with another user's selection handles?

Because of these reasons, we chose a different metaphor that appeared to be more appropriate for use in SDG applications. We developed the "local tool" metaphor that represents tools as separate icons that lie on the data surface along with user data [1]. The goal was to represent all of the applications functionality as tools.

For example, a user chooses a crayon tool for creating freehand lines. The tool has its own pen color and line width. Once the user has chosen the tool, he/she is the only person who can configure the tool to behave differently. By clicking the crayon on the color tool, the crayon's drawing color is changed (the tool changes color to provide feedback), and by clicking on the sharpen tool, the crayon's line width is modified (the tool tip changes size to provide feedback). In the tool approach, feedback is made simpler because each user's cursor is already an enlarged tool icon, whereas more traditional approaches to feedback in SDG systems have required the use of home areas or unattractive looking cursor constructs [3, 15].

KIDPAD

In order to test the Local Tools architecture, it was necessary to build a multi-user application. Our earlier work with building a collaborative drawing program for children indicated that it would be a rich source of potential conflict and interaction and it would likely provide a task that could benefit from SDG [7].

Our earlier work also showed the importance of using an iterative design process involving children as design partners when building applications for kids. We felt that iterative design would be even more important when building applications for co-present collaboration. Seventy-two children from a local elementary school helped design and test the KidPad application over a period of seven months. The starting application consisted of only three tools, one crayon for each user, and a shared eraser. By the end of the design process the kids had helped create over 20 different tools.

Since this was to be an application that enabled users to *collaboratively* create drawings, it was important to pay attention to how the application affected user relations and their collaborative behavior more than how efficiently drawings could be created [14, p. 95]. One such example is the evolution of the eraser tool. The original eraser was overly simplistic, it would erase the entire drawing. The kids quickly got frustrated not being able to erase small mistakes without the need to start over from scratch, and asked for a better eraser. The second generation eraser allowed users to erase individual line segments, instead of the entire drawing. Upon addition of the new eraser tool, however, the collaborative behavior of the groups changed fairly dramatically. They became overly critical of each other's work ("that's ugly, get rid of it") and they spent the majority of their time erasing.

A good solution to the problem of erasing took about three weeks of iterative testing, and involved the creation of three different tools: a bomb tool for erasing all of one users work; an eraser with two modes, a rub-out mode that would only erase lines drawn by the current user and a click mode that would erase any line clicked on; and a hand tool for picking up and moving lines. This combination of tools allowed users to easily erase simple mistakes, easily start over from scratch and erase big mistakes without affecting the other user, as well as move lines without having to erase and redraw them.

Usability Test

To evaluate the success of our ideas we conducted a usability test involving 60 students in the 3rd, 4th, and 5th grades of a Hawthorne Elementary School in Albuquerque, NM. Students were grouped into same-sex pairs, and were randomly assigned to either a single input device condition, or an SDG condition. We summarize the results of this study here.

Earlier pilot studies demonstrated the need to have a positive interdependence on collaborating partner's goal structures [22] as was indicated by Johnson and Johnson [12]. Therefore, the children involved in the study were told that they and their partner were a team in a design contest sponsored by the University of New Mexico. The researchers were building technology for kids and they wanted to know what kids thought about the technology they used, and what they wanted the technology of the future to look like. The teams were asked to complete a series of three drawings that would be entered as a team effort into the contest, in which all teams would compete against one another for prizes.

The children were given access to KidPad with one or two mice depending on the condition to which they were assigned to. They used KidPad once a week for 15 minutes during their regularly scheduled computer lab for a period of four weeks. At the final session they were given a switched condition: groups that had been using KidPad with only a single device were given multiple devices, and vice versa. All interaction with the application was logged, and each session was observed by the investigators as well as recorded on videotape for later verification of any observations. After the final session, the teams were given an informal verbal debriefing, to see how they felt about their ability to work together as a group in each condition. They were asked which condition they felt was the easiest to do drawings, which condition was the most fun, and which condition they would choose for use in other applications. Due to scheduling difficulties only 46 of the 60 students were able to complete the final session.

Results

One initially surprising result was the data for the post evaluation debriefing. We anticipated that the groups would be split as too which environment would be considered the easiest to use, a single input device or multiple input devices. Only seven children (15%) thought that one device was easiest to complete the drawings, while 37 (85%) felt the two device condition was easiest, and two children (4%) were undecided. Forty-five children (98%) answered that they felt that it was most fun using two devices. Only one child (2%) thought that one device was more fun. The answers for the final question (which condition kids would like to use for other computer applications) were identical to the answers for the first question (which condition was most fun). This suggests that having fun may be more important for kids than efficiency of task completion.

The children were also given the opportunity to say why they felt either condition was better. The one girl who preferred the one-input device condition did not say why. The others described why they preferred the SDG condition. A summary of the most frequent responses are:

Response	Frequency	Examples
No turn taking	49%	"We didn't have to share"
Parallel work	35%	"We can do different stuff at the same time"

Some comments worth highlighting follow. In response to our question of why they preferred SDG, one child commented "because there's two mouses!" Many of the kids thought it was obvious that two had to be better than one. Another said, "if [my partner was stuck and] I wanted to help there's another mouse" – peer-teaching was an advantage that even the kids were aware. One girl said, "[with two mice] you could do whatever you want" –KidPad didn't enforce collaboration, kids could work individually, if they chose.

The majority of kids (20 kids, 77%) who had used the two mouse condition complained loudly when they were only given a single mouse for the final session: "Hey! Where's the other mouse?" and "If there's only one mouse, I'm going back to work at my other computer", were typical reactions. The opposite reaction was common in groups that had only used a single mouse and were now given two mice: "Coool!" was nearly a unanimous response (18 kids, 90%). One girl didn't want to complete the final session because she was frustrated over having to share. When told she didn't need to share anymore her attitude changed, and she didn't want to leave the computer when their session was over.

CONCLUSION

This paper describes a model for co-present collaboration that we call Single Display Groupware. Several research groups have recently developed forms of SDG. We have tried to describe a framework for these projects to help understand common problems, and to suggest ways that technology developers should incorporate low-level support for SDG into their systems, so that the scenarios we introduced in this paper could become a reality.

The usability studies conducted to date, both by ourselves and others, have indicated that existing technology has a number of shortcomings when used for co-present collaboration. It appears that SDG technology enables new interaction modalities and can reduce some of the shortcomings observed with existing technology. It also may create new interaction problems. To better understand the overall impact that SDG technology can have, and to better design SDG applications, longer-term naturalistic studies are needed, and we hope that many people will continue to develop and evaluate SDG technologies and systems.

ACKNOWLEDGEMENTS

We would like to thank Angela Boltman and the children from Hawthorne elementary school in Albuquerque, NM for making the user study of KidPad possible. In addition, we appreciate the CHIkids at CHI 96 and CHI 97 who evaluated early versions of KidPad. Finally, this work could not have been done if it weren't for the other members of the Pad++ team, especially Jim Hollan and Jon Meyer. This work, and Pad++ in general has been largely funded by DARPA to whom we are grateful.

REFERENCES

1. Bederson, B. B., Hollan, J. D., Druin, A., Stewart, J., Rogers, D., & Proft, D. (1996). Local Tools: An Alternative to Tool Palettes. *In Proceedings of User Interface and Software Technology (UIST 96)* ACM Press, pp. 169-170.

2. Bederson, B. B., Hollan, J. D., Perlin, K., Meyer, J., Bacon, D., & Furnas, G. (1996). Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics. *Journal of Visual Languages and Computing, 7*, 3-31.

3. Bier, E. A., & Freeman, S. (1991). MMM: A User Interface Architecture for Shared Editors on a Single Screen. *In Proceedings of User Interface and Software Technology (UIST 91)* ACM Press, pp. 79-86.

4. Bricker, L. J. (1998). *Collaboratively Controlled Objects in Support of Collaboration*. Doctoral dissertation, University of Washington, Seattle, Washington.

5. Buxton, W. (1994). The Three Mirrors of Interaction: A Holistic Approach to User Interfaces. L. MacDonald, & J. Vince (eds.), *Interacting With Virtual Environments*. New York: Wiley.

6. Cockburn, A., & Greenberg, S. (1996). Children's Collaboration Styles in a Newtonian Microworld. *In Proceedings of Extended Abstracts of Human Factors in Computing Systems (CHI 96)* ACM Press, pp. 181-182.
7. Druin, A., Stewart, J., Proft, D., Bederson, B. B., & Hollan, J. D. (1997). KidPad: A Design Collaboration Between Children, Technologists, and Educators. *In Proceedings of Human Factors in Computing Systems (CHI 97)* ACM Press, pp. 463-470.
8. Greenberg, S., & Boyle, M. (1998). *Moving between personal devices and public displays.* Tech Report 98/630/21, Department of Computer Science, University of Calgary, Calgary, Canada.
9. Hall, E. (1968). *The Hidden Dimension*. Anchor.
10. Inkpen, K., Booth, K. S., Klawe, M., & McGrenere, J. (1997). The Effect of Turn-Taking Protocols on Children's Learning in Mouse-Driven Collaborative Environments. *In Proceedings of Graphics Interface (GI 97)* Canadian Information Processing Society, pp. 138-145.
11. Ishii, H., & Ullmer, B. (1997). Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms. *In Proceedings of Human Factors in Computing Systems (CHI 97)* ACM Press, pp. 234-241.
12. Johnson, D. W., & Johnson, R. T. (1991). *Learning Together and Alone, 3rd Edition*. Prentice Hall.
13. Krueger, M. (1991). *Artificial Reality II*. Addison-Wesley.
14. McGrath, J. E., & Hollingshead, A. B. (1994). *Groups Interacting With Technology*. Sage.
15. Myers, B. A., Stiel, H., & Gargiulo, R. (In Press). Collaboration Using Multiple PDAs Connected to a PC. *In Proceedings of Computer Supported Collaborative Work (CSCW 98)* ACM Press,
16. Pappert, S. (1996). *The Connected Family: Bridging the Digital Generation Gap*. Longstreet Press.
17. Pedersen, E. R., McCall, K., Moran, T. P., & Halasz, F. G. (1993). Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings. *In Proceedings of Human Factors in Computing Systems (InterCHI 93)* ACM Press, pp. 391-398.
18. Rekimoto, J. (1998). A Multiple Device Approach for Supporting Whiteboard-Based Interactions. *In Proceedings of Human Factors in Computing Systems (CHI 98)* ACM Press, pp. 344-351.
19. Shu, L., & Flowers, W. (1992). Groupware Experiences in Three-Dimensional Computer-Aided Design. *In Proceedings of Computer Supported Collaborative Work (CSCW 92)* ACM Press, pp. 179-186.
20. Smith, R. B., O'Shea, T., O'Malley, C., Scanlon, E., & Taylor, J. (1989). Preliminary Experiments With a Distributed, Multi-Media, Problem Solving Environment. *In Proceedings of First European Conference on Computer Supported Cooperative Work* Slough, UK: Computer Sciences House, pp. 19-34.
21. Stewart, J. (In Press). *Single Display Groupware*. Doctoral dissertation, University of New Mexico, Albuquerque, NM.
22. Stewart, J., Raybourn, E., Bederson, B. B., & Druin, A. (1998). When Two Hands Are Better Than One: Enhancing Collaboration Using Single Display Groupware. *In Proceedings of Extended Abstracts of Human Factors in Computing Systems (CHI 98)* ACM Press, pp. 287-288.