# Reducing Interference in Single Display Groupware through Transparency

Ana Zanella and Saul Greenberg
University of Calgary, Canada
*{azanella, saul}@cpsc.ucalgary.ca*

**Abstract.** Single Display Groupware (SDG) supports face-to-face collaborators working over a single shared display, where all people have their own input device. Although SDG is simple in concept, there are surprisingly many problems in how interactions within SDG are managed. One problem is the potential for *interference*, where one person can raise an interface component (such as a menu or dialog box) in a way that hinders what another person is doing i.e., by obscuring another person's working area that happens to be underneath the raised component. We propose *transparent interface components* as one possible solution to interference: while one person can raise and interact with the component, others can see through it and can continue to work underneath it. To test this concept, we first implemented a simple SDG game using both opaque and transparent SDG menus. Through a controlled experiment, we then analysed how interference affects peoples' performance across an opaque and transparent menu condition: a solo condition (where a person played alone) acts as our control. Our results show that the transparent menu did lessen the effect of interference, and that SDG players overwhelmingly preferred it to opaque menus.

## Introduction

Single Display Groupware (SDG) is a class of Computer Supported Cooperative Work (CSCW) applications that supports the work of co-located groups [Stewart, Bederson and Druin 1999]. The group shares the same display, which can be a large display or a monitor. Each member has his or her own input device, allowing all to interact simultaneously with the system. Figure 1 illustrates this, where we see two users, each with their own mouse, interacting simultaneously over a single monitor.

Zanella, A. and Greenberg, S. (2001) **Reducing Interference in Single Display Groupware through Transparency.** *Report 2001-683-06*, Dept Computer Science, University of Calgary, Alberta, Canada. February.

http://www.cpsc.ucalgary.ca/grouplab/papers/index.html

Figure 1. Two people in a Single Display Groupware situation, each with his own input device.

SDG provides its users with many potential benefits. Of course, SDG users can profit from the technological powers offered by the actual SDG application, which may be specialized to fit their task. SDG collaborators also gain the richness of face-to-face interactions for free because they are co-located: they can easily look at each other, see each other's gaze and gestures, have natural conversations, perceive each other's behaviour, and so on [Tang 1991; Whittaker and O'Conaill 1997].

Although these SDG benefits are self-evident, there is a surprising dearth of research in the area. One of the reasons for this deficiency is the difficulty of building SDG applications on personal workstations. These computers typically assume one user per workstation. Its top-level graphical user interface (GUI) provides only one text focus for the single attached keyboard, and one cursor for the single attached mouse. Even when we physically connect multiple keyboards and mice onto a workstation, the operating system just merges the device inputs into a single stream that is then passed onto the GUI. For example, when two users are moving their mice at the same time, the single cursor provided by the standard operating system will respond to both movements. Underlying programming languages and their graphical toolkits also provide poor or non-existent support for developers wishing to program SDG using multiple input devices. Events raised by keyboard or mice actions do not identify which keyboard or which mouse they came from. The standard graphical interface components—buttons, menus, list boxes, tool palettes and so on—are not designed to discriminate and respond to multiple users. This is disastrous for SDG, for by definition SDG users should be able to work simultaneously e.g., by raising and selecting from different menus at the same time. Similarly, SDG user actions must be treated separately e.g., people may be in different drawing modes as a consequent of selecting different colours from a palette, and each person's drawing actions should reflect this mode. The consequence of all this is that SDG designers and implementers often have to start from scratch. Device drivers recognizing multiple input devices must be written; programming languages must be

extended to discriminate input from multiple devices; and interface components must be totally redesigned if they are to respond efficiently to multiple users.

Even when the technical problems above are solved, there are other SDG usability issues that must be addressed. One specific interface issue we are investigating, and the focus of this paper, is *interference*: one person can raise an interface component (such as a menu or dialog box) in a way that hinders what another person is doing i.e., by obscuring another person's working area which happens to be underneath the raised component. Interference is a problem because it can distract and impeded SDG users from their tasks.

After first summarizing related work in SDG, we will describe interference in more detail. We will then suggest that transparent interface components may be a possible solution to interference: while one person can raise and interact with the component, others can see through it and can continue to work underneath it. Next, we will describe our implementation of a simple SDG game that we will use to test the efficacy of transparent SDG menus. In the subsequent sections, we will present our controlled experiment and our analysis of how interference affects peoples' performance when playing the SDG game using opaque *vs.* transparent menus: a solo condition (where a person plays alone) acts as our control. We close by describing the broader implications of our results to SDG design.

# Related Work

Bier and Freeman [1991] built one of the first SDG systems: a toy rectangle and text editor. They explored many SDG issues: how input devices are registered in the system; how multiple users are identified and 'attached' to particular devices; how different users can simultaneously manipulate the same data object; how individual mode information is captured and displayed; how multiple selections of data can be done; and so on. While a tour-de-force exposing many SDG issues and suggesting possible solutions, the authors did not, unfortunately, continue this line of research.

Several years later, researchers re-discovered SDG. Most of their efforts concentrated on showing that SDG systems could have a positive impact in educational settings involving children. Inkpen, McGrenere, Booth and Klawe [1997] studied how children share a single mouse *vs.* multiple mice when using a single display containing only one cursor. For the multiple mice situation, two types of turn-taking were tested to mediate access to the single cursor: *giving* (where one passes control to the other) and *taking* (where one takes control from another). In either situation, their results suggest that collaboration increased and that children had more fun when using multiple input devices to access the single cursor. Inkpen, Ho-Ching, Kuederle, Stacey and Shoemaker [1999] then explored the effectiveness of a true SDG setting, in this case testing pairs of children solving a puzzle in three conditions: a paper-based setting, a one-mouse / one-cursor setting, and a two-mice / two cursor SDG situation. Results indicate the advantage of the true SDG situation: most children preferred the two-mice / two-cursor situation since they could play together simultaneously, and they exhibited significantly

less off-task behaviour. Another research group based mostly out of the University of New Mexico also explored SDG use by children, in this case through an innovative SDG application called KidPad [Druin, Stewart, Proft, Bederson and Hollan 1997]. In particular, Stewart, Bederson and Druin [1999] studied how pairs of children collaborated when creating stories in KidPad. Each pair interacted together in either a one or a two-mice condition for three sessions, and then used the other condition for a last session. As before, children preferred the two-mice situation: they were more engaged in their task and they had more fun. Stewart, Raybourn, Bederson and Druin [1998] summarized several benefits they saw in true SDG: collaboration and communication increases, conflicts are reduced, and children offer and solicit help more often. Research in SDG use by children is continuing. For example, Hourcade, Bederson and Druin [2000] are now exploring how two children using a special-purpose SDG browsing tool can navigate to different parts of a shared library. Benford *et al* [2000] presents an enhanced version KidPad, where some tool functionalities are activated *only* when two children work collaboratively. For example, one child can draw with a basic colour, while two children can create a new colour by combining their colour tools.

Another thread of research considers how input devices other than the mouse can be used in a SDG setting. Myers, Stiel and Gargiulo [1998] explored personal digital assistance (PDAs) as input devices for SDG in their Pebbles Project. They described several advantages and disadvantages e.g., that PDAs can display output as well as input, that there are screen real estate problems associated with PDAs, and that PDAs can afford much more powerful interaction techniques when compared to a mouse. They created several applications demonstrating the capabilities of a PDA-based SDG system, including a shared editor, a scribble application, and a slide show system. Whereas the Pebbles project mostly used the PDA as an input device, Greenberg, Boyle and LaBerge [1999] considered how personal information created on the PDA could be brought into a face-to-face SDG setting, and how that information could then be manipulated on the PDA, the shared screen, or both. They were mostly concerned with the movement of personal information to a public space and back again, and concluded with a listing of problematic design issues that result from the distinctions made between personal and public information.

Next, researchers in Group Decision Support Systems have a long history of developing special purpose computer-augmented meeting rooms [Stefik et al 1987; Nunamaker et al 1992]. The room often has a very large display with its own connected computer. Participants usually have their own computer as well. A facilitator often controls the large display, and uses it to collect and show information gathered from each individual. Alternately, each person can switch his or her computer so that it appears on the large display. While related, these meeting rooms are a genre of their own; they are not quite single display groupware.

Finally, many game producers, such as Nintendo and Sony Play Station, have commercial SDG systems in every day use. Unlike standard commercial workstations, these specialized hardware boxes and many of the games often recognize up to four input devices and four players. The standard approach taken in most games is to split

the screen, where each player has their own dedicated portion that is a viewport into the virtual world. Unfortunately, there is little in the way of research reporting within this arena.

## Interference and Transparency

We now return to our description of *interference* in SDG. We define interference as the act of one person hindering, obstructing, or impeding someone else's view or actions on a single shared display. Interference arises when one person can raise an interface component (such as a menu or dialog box) over another person's working area. Because interface components are opaque, that other person cannot see beneath it, and are thus precluded from continuing their work. Any interface component that appears over the primary working area has potential to cause interference: pull down and popup menus, floating pallets, secondary windows, dialog boxes, and so on. These components may appear as a consequence of several activities: they can be raised directly by one user (e.g., by popping up a menu to make a selection: see Figure 3), or as a side effect of an action (e.g., a confirmation dialog box), or by the system itself (e.g., a system error or warning message appearing in a pop-up window).

The obvious solution of dedicating a portion of the screen real estate for displaying these components does not really work. First, this will lessen the available working area, which is a serious disadvantage because screen real estate is already very tight in SDG settings. Second, raised components can be quite complex and thus too large to fit e.g., a dialog box or palette with many options. Third, when there is one component type per person (e.g., when it is a component that displays an individual's mode), the number of components could increase with the number of collaborators.

Another solution is to do away with these floating and transient components altogether. For example, Druin et al [1997] proposed the idea of *local tools*, where large simple tools sitting directly on the work surface would replace traditional floating tool palettes. That is, local tools are guaranteed to appear within the space rather than above it. While reasonable for certain applications (Druin applied these to interfaces for children), we believe it cannot be generalized to all applications. For example, functionally rich applications may have so many tools and options that it would be unreasonable to map each to a simple tool; they would consume too much screen space.

Yet another solution is to display individual actions on the input device, as possible with PDAs. While promising, the problem here is that an individual's actions are now hidden from view. Thus other participants may no longer be aware of the actions that a person is taking because they cannot see them [Gutwin and Greenberg 1988].

Our own solution maintains the notion of floating and transient interface components while introducing the idea of making them semi-transparent. Transparency makes it not only possible to see the component itself but also what is underneath it. The effect is that when a semi-transparent component appears on top of a person's working area, that person is able to see through it and can continue his or her work. To allow the person to work underneath, each component responds only to its owner's inputs and passes all

other inputs to the underlying working area. For example, a semi-transparent menu raised by one person will let only that person select from it; intercepted actions of other people working underneath the menu will be passed on, thus allowing those people to continue their interactions with the underlying working surface.

Our own transparent interface components extend Harrison's previous work on applying semi-transparent interface components to single user applications [Harrison, Ishii, Vicente and Buxton, 1995; Harrison and Vicente 1996]. She was mostly interested in how a single person could use these components while still being able to see underneath them. She gave people menus with different degrees of transparency over various background textures, and then explored how well people could differentiate between these foreground and background layers: she found a reasonable compromise using objects that were 70% transparent. At the University of Calgary, researchers also used transparency, but this time as a way for a collaborating group to stay aware of one-another's actions in a distributed groupware setting [Greenberg, Gutwin and Cockburn 1996; Cox, Chugh, Gutwin and Greenberg 1998]. They used transparent overviews, where one user would see his or her detailed working area in one layer, with a transparent overview showing the entire workspace layered on the top of it. Their results also indicate that transparency is promising, particularly if used at the 70% level.

Because our SDG transparent components are quite different than these other uses of transparent components, we were uncertain as to whether they would help or hinder SDG collaboration. To help us judge whether transparent interface components are an effective way to mitigate interference effects, we built and tested an SDG version of a two-person game. In this game, one person would try to complete a task as quickly as possible, while the other person would intentionally try to interfere with the first one by raising a menu in their way. The next section will describe this system and its implementation, with subsequent sections detailing the study and our results.

# The Game

The SDG game used in the study is based on a "connect the dots" task. One player would try to draw a line connecting a series of numbered dots as quickly as possible, while the other player would try to slow down the first one by raising a menu in their way. The menu could be either semi-transparent or opaque.

A screen snapshot of a game session is illustrated in Figure 2. The player connecting the dots (using the pencil cursor) has connected dots 1-6, and has almost reached dot number 7. The other player (the arrow cursor) has raised a transparent menu over the first player.

We had three main implementation challenges when developing this SDG application. Although implementation of SDG is not the focus of this paper, we list these difficulties and how we solved them so that others wishing to replicate this (or similar) study can do so with less effort. These were:

- recognizing and treating multiple input devices,

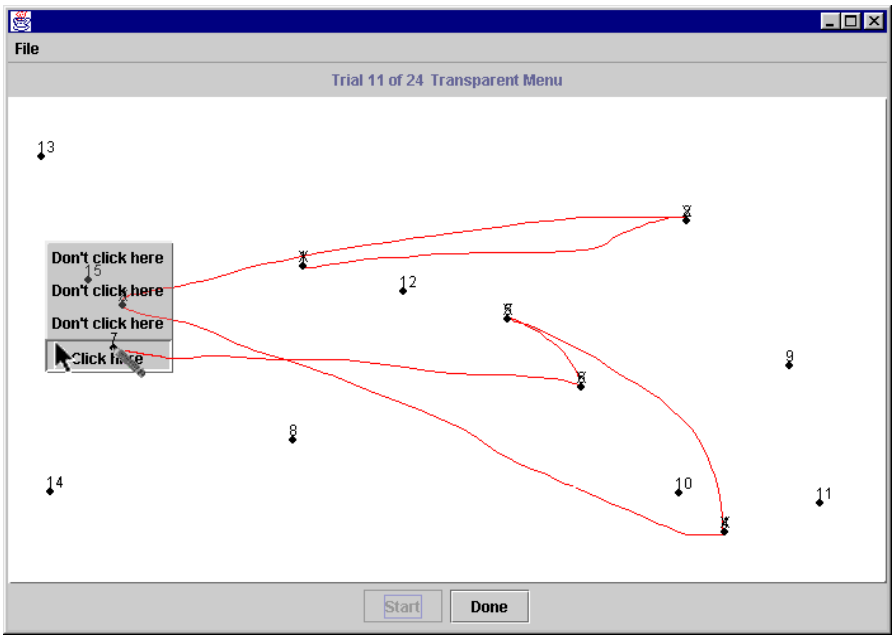


Figure 2 – A snapshot of the SDG connect the dots game. One player has raised a semi-transparent menu over the other player, who is working underneath it.

- designing interface components that respond appropriately to multiple input devices, and
- implementing semi-transparent interface components.

## Multiple Input Devices

As previously mentioned, conventional window and operating systems are not particularly adept at managing multiple input devices, particularly if they are mice. In many cases, it is up to the programmer to write device drivers that can interpret data generated by an input device attached to (say) a serial port e.g., as was done in MMM [Bier and Freeman, 1991].

Fortunately for us, Hourcade and Bederson [1999] developed MID—a dynamic link library and Java package running on Windows 98[1]—which implements an architecture that handles multiple Universal Serial Bus (USB) mice. MID extends the Java event mechanisms. In order to have access to multiple mice the standard Java events have to be replaced with the extended MID events. Programming with MID is very similar to programming with the Java events model. The main difference is that MID provides a unique mouse ID for each mouse seen by the system that can be retrieved when a mouse

[1] As far as we know, it is not possible to get separate input streams for multiple mice in Windows 2000, which insists on merging all mice input streams into a single one. This is a serious problem, and presents major technical obstacles for easy development of SDG systems. We have circumvented this problem somewhat by changing the USB chip on the mouse so that it presents itself to the system under a different name (we call it a Marmot). Because it is no longer recognized as a mouse, each Marmot input stream is available separately. Unfortunately, this also means that a separate Marmot driver must be installed and that program code must be tailored to recognize this special input stream.

event occurs. In this sense, it is possible to know which mouse triggered the event, and to treat it accordingly. Consequently, our game was written in Java and used MID.

## Interface Components that Recognize Multiple Mice

Each player can interact with the connect-the-dots game both simultaneously and independently of the other. Each player (and thus each mouse) is represented by its own cursor: a pencil for the first player who is connecting the dots, and an arrow for the second player who is raising the menu (Figure 2). Each player's actions are interpreted differently: a mouse press and drag by the first player draws a line, while for the second player it raises a menu and positions the cursor over an item.

To do this, we had to redesign and implement the interface components—the menu and the canvas—so they would recognize multiple mice. First, we assigned the drawing functionalities to $mouse_0$ and the menu functionalities to $mouse_1$. Second, we had to make these interface components respond only to their owner's input. A player using $mouse_0$ should not be able to select from the other player's menu, and should be able to continue to draw underneath that menu. Conversely the other player using $mouse_1$ should be able to raise a menu and make a selection from it while not affecting the drawing surface. Yet other shared interface components, such as the pull down menu and the buttons seen on the top and bottom of Figure 2, should respond to both players and consequently both mice.

While simple in concept, the problem is that no conventional widget set exists that recognizes multiple mice in this way[2]. This required us to completely re-implement the interface components to take mouse identification into consideration. For example, the SDG popup menu in Figure 2 is our own implementation, using a Java panel containing sub-panels, which in turn contains a label (the menu items). This meant that we were responsible for coding all visible effects corresponding to menu interactions, such as the raising of the menu and the highlighting of selected items. The component code also had to make a decision concerning each mouse event it saw. For example, when the menu received an event from $mouse_0$, we had to dispatch it to the drawing surface.

The necessity of redesigning interface components to handle situations such as these is one of the main obstacles to rapid SDG development. Quite simply, existing programming languages do not provide interface components that know how to deal with multiple inputs. This requires them to be redesigned from scratch, adding considerably to the burden of programming SDG systems.

## Transparency

The third implementation issue was implementing transparency on the popup menu. Fortunately, Java implements an alpha level for every colour; this can be adjusted to control the transparency level of the drawn object. Because we had already re-implemented the pop-up menus, it was fairly straightforward to specify the alpha level

---

[2] Widget development for SDG is now on-going in several research labs: our own at the University of Calgary; at Simon Fraser University [Shoemaker 2000]; and at the University of Maryland [Hourcade, Bederson, Druin 2000].

of its constituent components. For example, our semi-transparent popup menu uses an alpha level of 70 selected from a range from 0-255 to draw the panel and its sub-panels: this makes it slightly more than 70% transparent. However, we leave the text labels opaque for better readability[3].

While reasonable in our case, we recognize that other languages and graphical widget sets may not provide this ability to do alpha-blending. This could lead to significant implementation difficulties and/or performance penalties.

# User Study

We ran a controlled user study[4] in order to analyse the efficacy of semi-transparent popup menus when compared to opaque popup menus, using our SDG version of the "connecting the dots game". As we will describe below, we measured the level of interference that both types of menus create when users are doing their tasks as well as their levels of satisfaction. We focused the game towards the worst case of interference, i.e. where one user wanted to interfere with the other.

## Null Hypothesis

There is no difference in the *time* for a player to complete a connect-the-dots task or in their *menu preferences* (as measured by a questionnaire) when playing in a *solo* condition (i.e., by oneself) or in the *opaque* and *semi-transparent* blocking condition (i.e., when an opponent tries to slow down the player by raising a menu of a given type in their way).

## Subjects

We recruited and ran 30 pairs (60 subjects) in our study. Subjects were solicited from undergraduate and graduate programs at the University of Calgary. Subjects were asked to sign up in pairs, and as a result all but one pair were friends who knew each other. All appeared comfortable playing a competitive game with each other. All subjects were well-versed with computers, mice, and popup menus. When asked about familiarity with SDG systems, most answered that they had played multi-user videogames before. Each person was paid CDN$10.

## Materials

Our study situation used the SDG game and MID software as described previously running on Windows '98. Hardware included two USB mice, and a standard 1280x1024 19" display, and a modern PC. System performance was not an issue. The physical set

---

[3] Special fonts customized for transparent situations can be used instead [Harrison et al, 1995, 1996], but these were not necessary in our particular situation.

[4] A brief description and preliminary analysis of the study was reported in Zanella and Greenberg [2001].

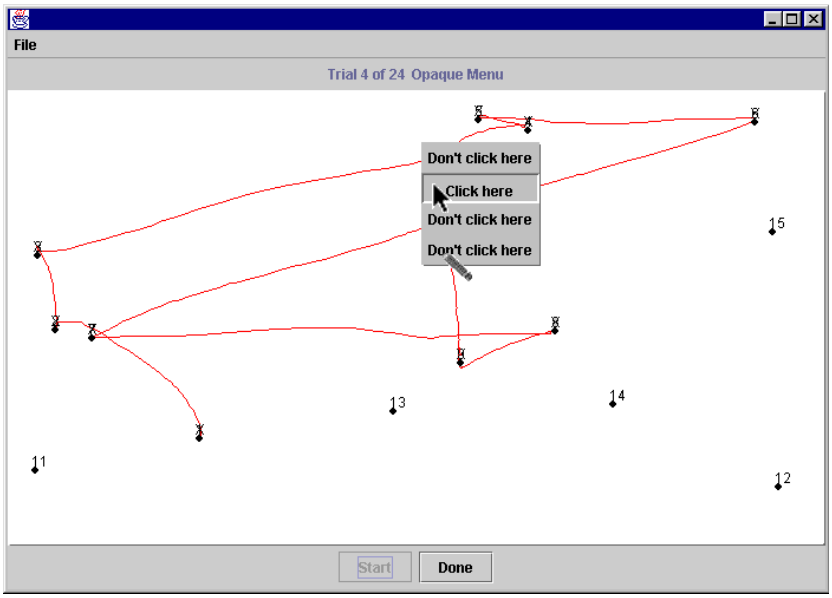


Figure 3 – The opaque menu version of the game. Notice that the player is 'blocked' by the interferer. While the player can still see the pencil cursor, she cannot see the next dot to connect, as it is located on the drawing surface underneath the menu.

up was similar to that seen in Figure 1, except that an observer was also seated behind both participants.

## The Task

One subject, who we called *player*, was asked to connect 15 dots in numeric order as fast as possible. The player did this by drawing a line from one numbered dot to its successor using a left mouse button press and drag, and then marking each dot with an 'X' after it was connected by clicking on it with the right mouse button. A pencil distinguishes the player's mouse cursor (see Figures 2 and 3).

The other subject, the *interferer*, was asked to interfere or slow down the player as much as possible by popping up a menu in a location that would obscure the player's view of where to go or what to do. The interferer could raise the menu in a given part of the display by right clicking over the desired position. The interferer was also instructed to quickly select the menu item labelled "Click here". This item was randomly positioned in the menu each time the menu was raised, as shown by the differences between Figures 2 and 3. These figures also show that the interferer's cursor is an arrow.

This was a competitive task. The player's goal was to connect all the dots as fast as possible, while the interferer's goal was to slow the player down as much as possible. To keep the game 'fair' for the interferer, we slowed down the player by requiring them to right-click each dot as it was connected. This mitigates those cases where the player is otherwise much faster than the interferer (which we saw in some of our pre-tests).

Similarly, we instructed interferers to select the 'Click here' menu option as fast as possible; this mitigates against them indefinitely blocking the player.

## Conditions

There were three different types of trial conditions in the test, where a trial consisted of a single connect-the-dots game.

*Solo:* the player connected the dots alone, without any interference. This is our control: we expect players will have their best performance in this condition, and we do not expect that they could better this time on average.

*Semi-transparent menus:* both player and interferer play, and the menus are semi-transparent (see Figure 2).

*Opaque menus:* both player and interferer play, and the menus are opaque (see Figure 3).

## Procedure

After they signed a consent form, we administered a pre-test questionnaire to each pair to collect information about their abilities with computer, mouse, popup menus and SDG systems. Each person in the pair was then randomly assigned to be either the player or the interferer. They kept these roles across all trials.

Each pair played 24 games divided into 8 sets. Each set contains the three different game conditions—solo, semi-transparent, and opaque—presented in randomized order. Each game displayed the 15 dots to be connected. As the dots were randomly repositioned for every game, no two games were identical for each pair. All pairs played the same games in the same order but in different conditions.

We considered the first set of three games as training trials, where players and interferers could explore the system and ask questions. We did not include these trials in the analysis.

For the remaining seven sets, we recorded the total time the player took to connect all the dots in a game. We also recorded the number of interferences as the number of time a popup menu was opened on the top of the player's immediate working area. While not part of the study hypothesis, this later data is used to check for situations where gross performance differences exist between the player and the interferer. We also observed the reactions, behaviours, expectations, comments and strategies of participants.

After playing all the games, the participants answered a post-session questionnaire that asked them about their menu preferences and how the menu types affected their tasks.

# Results

We watched all the pairs as they played. We wanted to observe their reactions, behaviours, expectations, comments and strategies.

All pairs engaged with the task. They appeared comfortable playing a competitive game. Interferers delighted in blocking the player's view, and both tried to trick each other by developing game strategies. All played in an appropriate manner, i.e. the player connected a dot before going to the next, all the dots were connected before starting a new game, the interferer was selecting the right option from the menu and did not leave the menu opened for a long period of time, etc.

## Performance

We analysed how long the player took to connect the dots across the different trial conditions. As mentioned previously, this gives an indication of the efficacy of each menu type as contrasted to each other and to the solo control. We collapsed the data within each pair into an average time / condition type. To get a sense of this data, we first compared how each pair faired over these conditions. In almost all cases, the average within-subject time relationships when performing these conditions are: solo < transparent < opaque. A single factor ANOVA shows that these differences are statistically significant ($F= 16.36$, $p<0.05$). A post-hoc t-test shows statistically significant differences between every condition, as summarized Table 1. Thus the null hypothesis is rejected. Figure 4 illustrates these differences by displaying the average performance time and standard deviation to complete a game for all subjects in each condition.

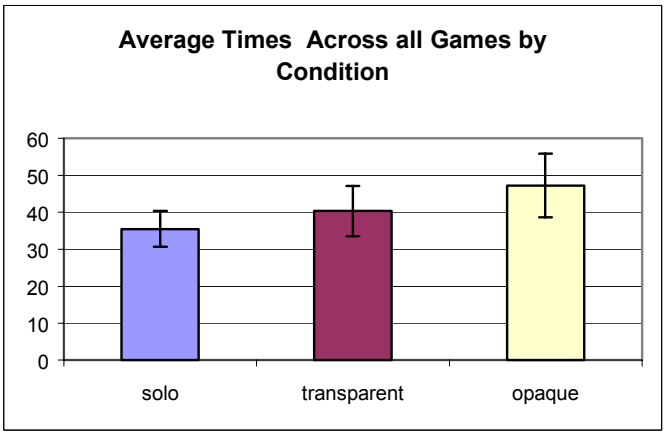


Figure 4 – Average performance time to complete a game for all subjects by each condition. Standard deviation is shown in each bar.

| Condition | P(T<=t) two-tail |
|---|---|
| Transparent *vs.* Opaque | 0.0017 |
| Solo *vs.* Opaque | 0.0000 |
| Solo *vs.* Transparent | 0.0108 |

Table 1. Post-hoc t-test analysis

We performed a few other analyses to look for any hidden effects that could have influenced our results. First, it is entirely possible that people's performance changed over time, perhaps due to learning or fatigue. We analyse each trial type separately, where we calculated the average time for completing a particular game in a particular trial. Results are graphed in Figure 5.

In this graph, we do see a small increase in performance time over the first few games. This is likely a learning effect, where people are getting used to the mechanics of playing i.e., which mouse button to click, how to search for the next number and so on. However, what is immediately obvious by visual inspection is that the average time to complete a particular game is still solo < transparent < opaque. That is, it is unlikely that the statistical differences seen in our analysis are confounded due to some relative performance change in the game over time.

We calculated the average number of interferences per game in each condition, to analyse the relation between interference and performance. While the graphic suggests that there are differences between interference levels in the opaque and transparent conditions, a single factor Anova shows that these differences are not statistical significant ($F=4.6069$, $p=0.53$). We do see a minor increase in the number of interferences on the first half of the games, likely a learning effect. The decrease in the

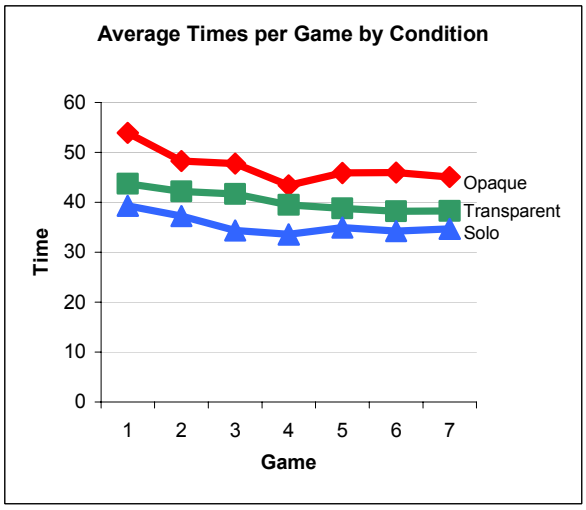


Figure 5 – Average time for players to complete a particular game in a particular trial.

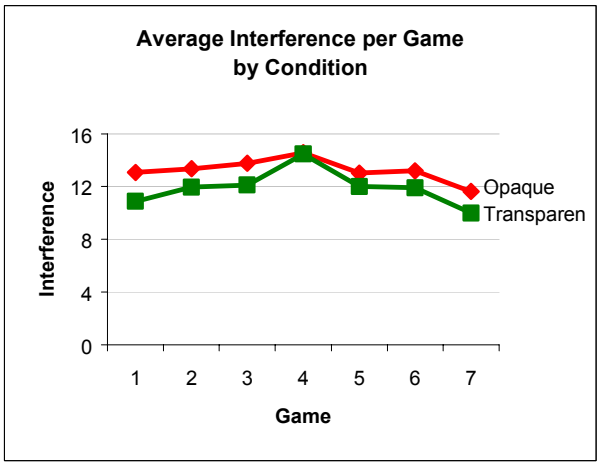


Figure 6 - Average number of interferences per game in a particular trial type.

end is probably a result of minor user fatigue.

Although we see no statistical significant difference on interference levels between opaque and transparent menus (i.e., the number of interference was similar in both situations) there are differences in performance in both situations. These results lead us to conclude that the opaque menus create more interference than the semi-transparent menus. However, semi-transparent menus do exhibit interference effects as performance is not quite as fast as in the solo control condition.

## Preferences

Through our post-session questionnaire, both the player and the interferer stated their opinions and preferences in terms of how the menus affected their task.

Using a five-point scale, with opaque on one side and transparent on the other, we asked subjects which type of menu they preferred in the SDG situation (i.e., without regard to their player or interferer role). Their responses strongly indicate a preference for transparent menus over opaque ones, as illustrated in Table 2. 34 of the 60 subjects strongly preferred transparent menus, and 9 more had a weak preference. Only 10 of the 60 liked the opaque menus.

Using a three-point scale, we then asked subjects how the different menus affected their particular task when acting as player or interferer. In these responses, tabulated in Table 3, almost all players thought that transparent menus made it easier for them to continue their work in spite of interference (28 of the 30 players). On the flip side, almost all interferers thought that transparent menus made it harder for them to interfere with the player (25 of the 30 interferers). While these results are not analyzed statistically, they obviously enforce our rejection of the null hypothesis.

## Qualitative Observations

We watched all pairs as they played in all conditions, and interviewed them afterwards. We saw that the pairs quickly engaged with the game, and became very competitive over time.

It was obvious that players greatly preferred the transparent to the opaque menus because it was easier and faster for them to connect the dots. As one player commented:

> "… After a while I did not even see the transparent menus anymore, it was like I learnt how to ignore them…"

Players become frustrated when they were blocked with an opaque menu. As one player exclaimed to an interferer who was taking their time selecting the item from the opaque menu: "Can you make your selection faster?"

As one would expect, the interferers preferred playing in the opaque over the transparent menu condition because it helped them block the player. This exchange, occurring at the beginning of an opaque menu game, highlights how a one pair's reaction:

*Interferer:* "I am going to bug you now!"
*Player:* "I hate these opaque menus!"

One specific pair (a boyfriend who was a computer scientist and girlfriend who was not) was very competitive. As a player, she celebrated every time a dot was connected under the transparent menu. As the interferer, he was noticeably excited when the opaque menus came up, and kept making fun of her when the popup menu blocked her view. In the end of the game, she playfully asked him to give her all the money he had earned; because she had done very well in spite of his merciless teasing, she felt she deserved it.

In informal post-test interviews, one subject said he really liked the semi-transparent menus in SDG, and that he would also want them even in a single user application. He explained:

> "Sometimes when you are making a search on [Microsoft] Word the result is positioned near the *find window*, so you have to move the window if you want to see the text related to the search. [Similarly,] the window to format text, to change a colour or font type, usually covers the text you are modifying … sometimes you open the window and you forget if you selected the right text."

We also observed that most players moved very quickly when playing in the opaque menus situation, for they wanted to minimize the actual times that the interferer raised the menu over their position. Although we told them to connect the dots as fast as they could, the players appeared more relaxed in the solo situation since they knew no interference would happen. We saw a similar relaxed attitude in the semi-transparent situation after players played a few games, probably because they knew they could still continue their job in spite of the interferer's best efforts. While this apparent speed-up in the opaque menus could have confounded our results, we still see that, on average, it took longer for players to complete games with the opaque menus (Figure 4). That is, the differences between conditions still exist *in spite of* the player's best effort to overcome the interferer.

We also saw that most of the pairs developed strategies of play after a few games. At first, interferers moved their cursors by the next point to be connected, and then popped up the menu when the player arrived at that spot. After some time they realized they were sometimes helping the player, as it showed the player where the next point was. To offset this, the interferer moved their cursors around while waiting, or just following the players' cursors before 'pouncing' on them. Some of the players tried to counteract this by first moving their cursors away from the correct dot (in order to 'trick' the interferer). An instance of this situation is illustrated in Figure 7: The player has just finished connecting and marking dots 1 through 5 (marking is shown by the 'X' on these dots), and then initially moved his pencil cursor away from dot 6 before quickly moving it back to dot number 6. When this strategy worked, the interferer was unable to

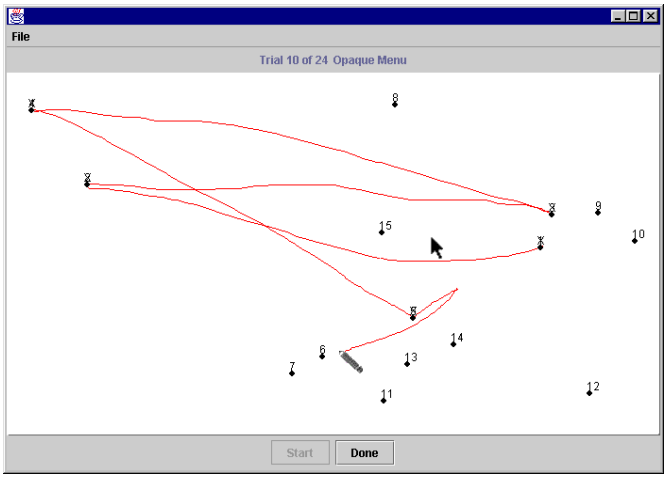


Figure 7 – In this game, the player successfully 'fakes out' the interferer by first moving away from the desired point and than rapidly switching directions.

respond as quickly as normal. However, the relatively flat performance curve over all games (displayed in Figure 5) suggests that these opposing strategies counterbalanced each other over time, and they likely did not confound our results.

# Discussion

Our results suggest that semi-transparent interface components can mitigate interference in Single Display Groupware. This is promising indeed, for it means that the existing genre of popup components (e.g., menus, windows, dialogue boxes, floating palettes) can be adapted to SDG, and that people can use these well-known techniques to interact with SDG systems. The only real difference is that users can see through them, and that they have to understand what component is their own and which belongs to others. From our observations, we saw that people quickly adapted to transparent components, and had no problem manipulating them or working underneath them. While 'standard' interface components would have to be redeveloped in order to work within SDG and to display themselves semi-transparently, the basic interaction technique remains the same. Ideally, transparency in SDG is only a programming issue rather than an interface issue.

However, we recognize that the situation we tested was simple, and that there is a danger of over-generalizing our results to all SDG situations.

First, we used only two users in a very controlled situation, and we are uncertain about what would happen if three or more collaborators were interacting simultaneously. For example, it may be possible for several semi-transparent components to be raised atop each other.

Second, we tested the worst case of interference, where one user intentionally tried to interfere with the other. Actual interferences in every-day SDG situations are probably far less numerous. If good feedback were provided to collaborators about what others intended to do, social protocols would likely lessen the number of actual interferences. People are, in fact, quite adept at informing others about possible conflicts and at mediating turn-taking when contention is unavoidable. Still there are times that collaborators in SDG cannot avoid interference. For example, one person may popup up a menu or dialog box without realizing that others would be affected. Or the system may have to raise a large error window, but there may be no place to position it that would not cause interference. Even if people do mediate their actions by resorting to turn-taking, we suspect that this sequential rather than simultaneous access to the space will lessen the amount of collaborations and people's feeling of satisfaction [Inkpen, Ho-Ching, Kuederle, Scott and Shoemaker 1999].

Third, we used a 1280x1024 resolution standard monitor as our shared display. Yet the probability of interference may decrease for larger, higher resolution screens (because people have more space to do their work), and increase for smaller ones (because people will likely contend for the same area).

Fourth, the interface component we tested—the menu—is fairly small and usually does not stay long on the screen. Larger and longer-lasting interface components, such as a dialog box, could create more interference problems to users. For example, a 'save

as' dialog box is quite large, and it often takes considerable time for a person to find the right folder and type the name of a file. In these cases, transparency could be even more helpful.

Fifth, our game used a foreground and background conducive to transparency. Excepting the drawing marks and the numbered dots, the background was fairly sparse. Thus it was easy to separate visually the text of the menu from the background objects. As Harrison et al [1995, 1996] noticed, backgrounds rich in visual information—pictures, contrasting colours, dense text—may make the visual separation of the layers difficult. Similarly, complex foreground objects may be difficult to separate from the background e.g. the many fields of a complex dialog box. While there are a few design techniques within transparency that help make certain items stand out, these are still in their infancy [Harrison et al 1995, 1996].

In summary, our transparency approach is successful in our test conditions, and we believe they are promising as a way to minimize interference in SDG applications. Users reacted positively to the semi-transparent popup menus, mentioning that the idea could also be applied to other widgets and even non-SDG settings. Still, we recognize that real-world factors can that both increase or decrease the benefits of this technique. To truly understand these factors and their effects, we need to develop serious SDG applications, deploy them into real situations, and study what happens.

# Conclusions

There are many issues involved in SDG development. Some are technical, for example, how multiple input devices are seen by the operating system and how programming languages support them. Other problems are related to the design of interface components that are adequate for several users sharing the same screen, such as recognizing multiple users' input and responding accordingly to each input.

In our study we investigated interference as one particular interface problem in SDG. We offered semi-transparent interface components as a way to mitigate interference effects. We then created a 'worst case' of interference, where one person intentionally tries to interfere and slow down another person by blocking them with pop-up menus. As our test results show, our approach of using transparency is appropriate for dealing with interference in our SDG situation. Although our setting was somewhat simplistic, we believe the idea of transparency could be generalized to a certain extent to other SDG applications.

## Acknowledgments

Thanks for Ben Bederson and Juan-Pablo Hourcade from the University of Maryland for graciously allowing us to use their MID software. We also thank our research participants for participating in the study, as well as all Grouplab researchers for their input. We are grateful to Microsoft Research, the Alberta Software and Engineering Research Consortium (ASERC), the National Sciences and Engineering Research Council of Canada (NSERC) and Smart Technologies who provided funding, some equipment, and encouragement.

# References


Benford, S., Bederson, B., Akesson., K., Bayon, V., Druin, D., Hansson, P., Hourcade, J., Ingram, R., Neale, H., O'Malley, C., Simsarian, K., Stanton, D., Sundblad, Y., and Taxen, G. (2000) Designing storytelling technologies to encourage collaboration between young children. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'00)*, Pages 556-563, ACM Press.

Bier, E. and Freeman, S. (1991) MMM: A user interface architecture for shared editors on a single screen. In *Proceedings of the 4th annual ACM Symposium on User Interface Software and Technology*, Pages 79-86, ACM Press

Cox, D., Chugh, J., Gutwin, C. and Greenberg, S. (1998) The usability of transparent overview layers. In *Summary Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'98)*, Pages 301-302, ACM Press.

Druin, A., Stewart, J., Proft, D., Bederson, B. and Hollan, J. (1997) KidPad: a design collaboration between children, technologists, and educators. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (*CHI'97),* Pages 463-470, ACM Press.

Greenberg, S., Boyle, M. and LaBerge, J. (1999). PDAs and shared public displays: making personal information public, and public information personal. *Personal Technologies*, Vol.3, No.1, March, Pages 54-64.

Greenberg, S., Gutwin, C. and Cockburn, A. (1996) Using distortion-oriented displays to support workspace awareness. In Sasse, A., Cunningham, R. and Winder, R. (ed) *People and Computers XI,* Pages 299-314, Springer-Verlag.

Gutwin, C. and Greenberg, S. (1998) Design for individuals, design for groups: tradeoffs between power and workspace awareness. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'98),* Pages 207-216, ACM Press.

Harrison, B. and Vicente, K. (1996) An experimental evaluation of transparent menu usage. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'96)*, Pages 391-398, ACM Press.

Harrison, B., Ishii, H., Vicente, K. and Buxton, W. (1995) Transparent layered user interfaces: an evaluation of a display design to enhance focused and divided attention. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'95)*, Pages 317-324, ACM Press.

Hourcade, J. and Bederson, B. (1999). Architecture and implementation of a Java package for multiple input devices (MID). Report HCIL-99-08, Computer Science Dept, University of Maryland, MD USA.

Hourcade, J., Bederson, B. and Druin, A. (2000) QueryKids: a collaborative digital library application for children. *Workshop on Shared Environments to Support Face-to-Face Collaboration at the ACM* CSCW'00, Dec. 2. http://www.edgelab.sfu.ca/CSCW/shared_environments.html

Inkpen, K., Ho-Ching, W., Kuederle, O., Scott, S and Shoemaker, G. (1999) This is fun! We're all best friends and we're playing: Supporting children's synchronous collaboration. In *Proceedings of the ACM Conference on Computer Supported Collaborative Learning (CSCL'99),* Pages 252-259, ACM Press.

Inkpen, K., McGrenere, J., Booth, K., and Klawe, M. (1997). The effect of turn-taking protocols on children's learning in mouse-driven collaborative environments. In *Proceedings of Graphics Interface (GI'97)*, Pages 138-145, Morgan Kaufmann Publishers.

Myers, B., Stiel, H. and Gargiulo, R. (1998) Collaboration using multiple PDAs connected to a PC, In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW'98)*, Pages 285-294, ACM Press.

Nunamaker, J., Dennis, A., Valacich, J., Vogel, D. and George, J. (1992) Electronic meeting systems to support group work. In Baecker, R. (ed) *Readings in computer supported cooperative work*. Pages 718-739, Morgan Kaufmann Publishers.

Scott, S., Shoemaker, G. and Inkpen, K. (2000) Towards seamless support of natural collaborative interactions. In *Proceedings of Graphics Interface (GI'00).* Pages 103-110, Morgan Kaufmann Publishers.

Shoemaker, G. (2000) Supporting private information on public displays. In *Extended Abstracts of the ACM Conference on Human Factors and Computing Systems (CHI'00),* Pages 349-350, ACM Press.

Stefik, M., Foster, G., Bobrow, D., Kahn, K., Lanning, S. and Suchman, L. (1987) Beyond the Chalkboard: computer support for collaboration and problem solving meetings. *Communications of the ACM*, 30(1), Pages 32-47, ACM Press.

Stewart, J., Bederson, B. and Druin, A. (1999) Single display groupware: a model for co-present collaboration. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'99)*, Pages 286-293, ACM Press.

Stewart, J., Raybourn, E., Bederson, B. and Druin, A. (1998) When two hands are better than one: enhancing collaboration using single display groupware. In *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems (CHI'98)*, Pages 287-288, ACM Press.

Tang, J. (1991) Findings from observational studies of collaborative work. In Greenberg, S. (ed) *Computer support cooperative work and groupware*. Pages 11-26, Academic Press.

Whittaker, S. and O'Conaill, B. (1997) The role of vision in face-to-face and mediated communication. In K. Finn, A. Sellen and S. Wilbur (ed) *Video-Mediated Communications*. LEA Press.

Zanella, A and Greenberg, S (2001) Avoiding interference through translucent interface components in single display groupware. In *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems (CHI'01)*, Short Paper, ACM Press.

Author's address

Ana Zanella
University of Calgary
Department of Computer Science
2500 University Drive NW
Calgary – T2N 1N4
Canada
Phone: +1 403 220-7686
Fax: +1 403 284-4707
azanella@cpsc.ucalgary.ca

Saul Greenberg
University of Calgary
Department of Computer Science
2500 University Drive NW
Calgary – T2N 1N4
Canada
Phone: +1 403 220-6087
Fax: +1 403 284-4707
saul@cpsc.ucalgary.ca