MULTI-USER SCREEN SHARING IN MULTI-DISPLAY GROUPWARE

Choon Jin Ng
Tunku Abdul Rahman University-College

Masahiro Takatsuka
The University of Sydney

Tong Ming Lim
Tunku Abdul Rahman University-College

ABSTRACT

Ever since the idea of computer supported cooperative work (CSCW) was first coined, screen sharing technology has been playing a crucial role in groupware for fostering an environment which enables people to share and exchange information more freely. This is known as a form of What-You-See-Is-What-I-Saw (WYSIWIS) collaboration. Screens can be shared within a group of people in the same room or remotely. As computers are becoming affordable, it is not uncommon to have a collaboration session involving multiple machines. However, existing groupware do not support simultaneous sharing of multiple machines' displays. This limits conference participation in the sense that multi-user participation is confined to only a single machine. This paper intends to present a multi-display groupware (MDG) architecture to make sharing multiple machines' screens possible while enabling true multi-user interaction. This architecture can be configured to be used in co-located, distributed and mixed-presence environment.

KEYWORDS

Multi-display groupware, WYSIWIS, remote collaboration, CSCW, Mixed-presence groupware.

1. INTRODUCTION

In a computer aided collaboration, being able to share screen content is crucial. Many remote collaboration done today consists not of one but a group of participants located at every site. However, current screen sharing system can only either support a distributed or a co-located environment. That is, screen sharing can only be done remotely between one-to-one participant, or locally between a group of participants. None of them support both environments simultaneously. With computers and monitors becoming increasingly affordable, many home users and organisations can afford several machines with multiple displays. This resulted in the proliferation of various groupware exploiting the availability of these hardware.

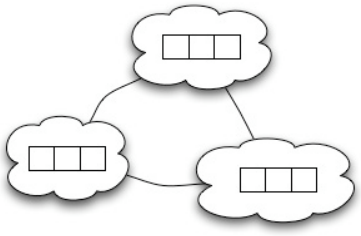

Figure 1. Screen sharing in a mixed-presence multi-display environment. Each cloud represents a site with three machines.

Therefore, a groupware for supporting screen sharing in both co-located and distributed environment is important. Such groupware is known as a mixed-presence groupware (MPG) (Tang, Boyle and Greenberg, 2004). Figure 1 shows how we can extend screen sharing support into a mixed-presence multi-display environment. The MDG system comprises of multiple displays shared by multiple users. There are three main challenges posed for such system:

1. Sufficiently integrate display and input control of shared screens in order to create a seamless workspace. The spatial order of displays must be preserved within the local and distributed site. Then, controls such as keyboard and mouse must be unified for seamless navigation.
2. A central architecture for supporting an arbitrary machine configuration for screen sharing.
3. Supporting multi-user inputs in current contemporary operating systems (OS) which inherently only support single-user interaction.

2. RELATED WORK

A single-display groupware (SDG) application uses a single display shared by multiple people (Myers, 1999). Naturally, the environment in which the SDG application operates is called a single-display environment (SDE). Likewise, a multi-display groupware (MDG) operates in a multi-display environment (MDE). In this literature, our definition of MDE comprises of multiple displays over multiple machines shared by multiple users. It does not include the usage of multiple displays over a single machine.

The benefits of MDE are well publicised (Robertson *et al.*, 2005; Biehl *et al.*, 2008). Current MDE research focuses on information organization through specific graphical user interfaces (GUIs) (Jeong *et al.*, 2006; Biehl *et al.*, 2008; Sakurai *et al.*, 2008), interaction technique (Davis and Chen, 2000), scalability and performance (Jeong *et al.*, 2006). However, there is virtually no research in the area of screen sharing in a MDE environment. Most of the existing systems are confined to either a co-located collaboration or a distributed collaboration, but not mixed-presence collaboration.

A collaborative workspace is one public workspace which can be shared to work on problems or to share information. However, Tse et al. found that interference can occur when more than one person is working in close proximity (Tse *et al.*, 2004). To avoid this, spatial separation and partitioning can be employed by partitioning a workspace into personal private space and group space. Then, during a collaboration session, a person can choose to work a sub-problem on his own. The public space can serve as a place to integrate solutions and to exchange information between collaborators. In SDG, if a screen is divided among several participants for personal use, the lack of space not only reduces participant workspace, but it also reduces privacy as participants need to work closer together and this can intrude other participants' privacy (Gutwin and Greenberg, 1998; Booth *et al.*, 2002).

Apart from workspace, the coordination of input controls are also crucial. In this paper, a system control consists of a set of mouse and keyboard inputs supplied on a per user basis. In a co-located environment, the CPNMouse (Westergaard, 2002) can be expanded with several displays while each user is allotted with his or her individual mouse control. However, having the case of multiple screens through multiple machines still poses some challenges. Since each machine has its own separate control system, the controls must be sufficiently integrated to support the notion of single workspace. Several existing collaboration systems such as the Mighty Mouse (Booth *et al.*, 2002), CPNMouse, Synergy, 3D Server (Sakurai *et al.*, 2008), and IMPROMPTU (Biehl *et al.*, 2008) do not allow true multi-user control experience. They either time-slice a cursor between multiple users or uses some kind of floor control.

On the other hand, MPX (Hutterer and Thomas, 2007) enables true multi-user experience by directly supporting multiple mice at system level. It maintained legacy application support. All multi-cursor events are sent directly to the application instead of time-slicing a single OS cursor. It is however tightly tied to the X window system.

The middleware GLIA allows collaborative application development to adopt a multi-cursor approach to co-located collaboration (Yuizono, Nishimura and Munemori, 2006). In GLIA, each user has a mouse cursor and a keyboard. These cursors are managed by its respective mouse agent which can be transported across different machines. A mouse monitor resides at the machine where the physical mouse/keyboard is located. Together with multi-cursor support, this system can support simultaneous work in single co-located environment.

3. MULTI-SCREEN SHARE ARCHITECTURE

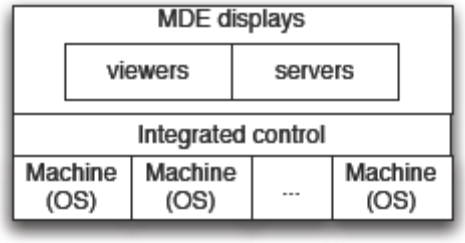

Figure 2. The Multi-Screen Share (MSS) architecture.

We have created the Multi-Screen Share (MSS) architecture capable of supporting multi-machine screen sharing in a mixed-presence environment. Figure 2 illustrates the MSS architecture where each site can consists of one or more machines. Cursor communications between machines are peer-to-peer and can migrate across different machines. An aggregation of displays from each machine then forms a MDE display. The display can belong to one of these two components: viewer and server. The server shares its contents to other remote machines while the viewer displays the shared contents.

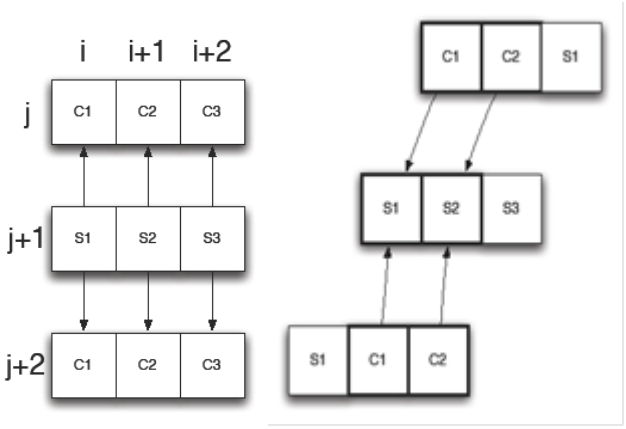

Figure 3. Illustration of MDE screen sharing with (a) strict view –left and (b) mixed view - right.

The MDE displays layer is responsible for streaming screen output to the network and receiving them for display. It enables flexible layout through the choice of configuring screens as strict view or mixed view.

A strict view refers to all sites having exactly the same content with the same number of screens arranged in the same position. Hence the collaboration is centred on only one set of workspace. To set up a strict view, for all sites, only one machine at a position can be a server while the rest must be clients. To clarify more, using the example in Figure 3, take i as a position of screen and j as the site number, then X(i,j) is the i^{th} screen position located at site j. Hence for a known position k, X(k,j), only one of the site in j is the server and the rest must be client. On the other hand, for a mixed view, it not only has the same attribute as strict view, but there are additional screens where they are not shared in order to keep them private. However, these screens' controls are still linked together. Hence this allows a single public with one or more private workspaces to be formed.

Again, in Figure 3, the strict view is the server set {S1, S2, S3} placed side by side and the controls are combined as a single workspace. The clients then replicates each server's view in an exact configuration. In a mixed view, the servers {S1, S2, S3} are placed side by side. However, {S3} is not shared to other sites. The client {C1, C2} connects to {S1, S2} respectively to form the public part of the workspace. The screen {C1, C2, S1}, {S1, S3, S3} and {S1, C1, C2} are then aggregated respectively to form a public-private workspace where the controls can be migrated smoothly across different workspaces. A noteworthy fact is that on both

view configuration, a mix of clients and servers are possible at any site and the control transition should remain smooth.

At the next MSS level, an integrated control layer is featured to ensure controls across all machines are regulated to produce a seamless workspace. There are two components in the integration layer:

- *Multi-cursor system:* Ensures each physical mouse plugged into any of the machine is detected. This is so that a new control (mouse and keyboard) can be issued to a participant. This layer also determines which machine the control is multiplexed. Therefore, it must be spatially aware of the position of its neighbouring machines.
- *Multi-user input system:* Having multiple cursors for each participant is not enough. When users are performing actions on the same machine simultaneously, the OS must reflect this condition and not simply serialize the commands. We were able to avoid the effects of time-slicing cursor by introducing the "direct-application event injection" mechanism. This mechanism will be described in a later section.

In a mixed-presence environment, sharing multiple screens among several sites consumes tremendous bandwidth. In the worst case of all sites transmitting its own screen at once, the total transmission done is n(n-1) times if unicast connection with n nodes is used. However, if multicast network is employed, then total transmission is reduced to a constant of only n times. Therefore, a multicast version of VNC called MVNC (Ng and Takatsuka, 2009), is incorporated into the MSS system.

To prevent spatial confusion, the positioning of screen should be the same relative to each other regardless of any site. Hence, participants at each site should have its screen position predetermined beforehand.

Lastly, an accompanied screen sharing application, named the M3DESKSHARE, was implemented to demonstrate the MSS architecture.

4. THE MULTI-CURSOR SYSTEM

When sharing screens, different cursor distribution strategy is required for different MDE configurations:

- *Single public space:* Within a local site, cursors should be transferable from one machine to another seamlessly. The cursors are then mirrored to other remote sites.
- *Private-public space:* In addition to the single public space requirements, cursor permission must be enforced to prevent cursors in one private space from intruding other private spaces. Only the mouse cursor in the public space is allowed to be mirrored.
- *Large tile display:* Large tile display usually resides in a public space for centralised monitoring. It aggregates a set of remotely shared screen. Figure 4 demonstrates three standalone machines A, B and C are to have their screen aggregated into a large tile display forming A', B', and C'.

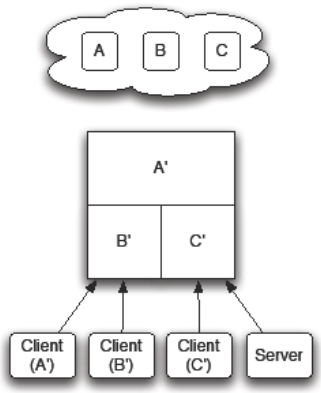

Figure 4. Large-tile display aggregates display from remote host A, B, and C. At the large tile display site, the remote screens are displayed through the clients and each client can access the displays merged as a single tile display. The server does not share any screen but merely plays a role as a controller.

The GLIA framework already provides a way to support multi-user collaboration on a set of machines in a single co-located environment. However, to support user control in a mixed-presence MDE environment, the following extensions to the GLIA are required:

- Each site will require its exclusive unit of GLIA environment.
- The agent and monitor behaviour needs to be modified to support this new environment.

To accomplish the abovementioned extensions, the concept of "control mirroring" is introduced so cursors at each site can be mirrored to other distributed sites. This concept is shown in Figure 5 with a purported mirroring between two units of GLIA environments. Each mouse is monitored by a mouse monitor. The management of each cursor and its events are linked to the mouse monitor via the mouse agent. Mirroring is performed when one agent is duplicated and is migrated to a remote target machine.

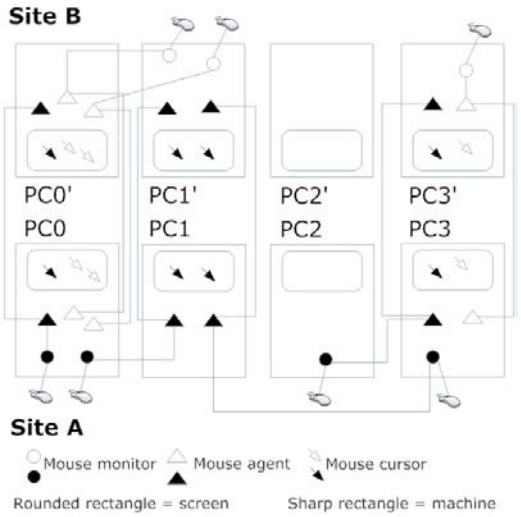

Figure 5. A mirrored GLIA framework between two sites, site A and site B.

At each MSS initialisation, the host will wait for any incoming agent migrated from other hosts or any new physical mouse detected locally. Any of these two events will trigger the creation of a mouse agent. From here, it is worth to note that a mouse agent can originates from three different sources:

1. *Local physical mouse:* A physical mouse is plugged into the host.
2. *Remote mouse agent within the same unit GLIA environment:* A mouse agent migrates from one host to another (within the same unit GLIA environment) as its cursor moves from one machine's screen to the other machine's screen.
3. *Remote mouse agent from different unit of GLIA environment:* This is the mirrored cursors from other GLIA environment unit.

For case (1) and (2), the agent is simply duplicated and mirrored to all the hosts' mirrors. However, if case (3) is exercised, the host will have to decide if it has interacted with the incoming agent's machine previously. If it had not, that implies that particular host has no knowledge of any existing external cursor. Therefore, the server will now need to mirror all agents it has in order to initialize the new incoming host. Once that host is initialized, any subsequent new agent from that host will only result in an update to all other connected mirrors.

As mouse activities are performed, the mouse monitor will generate events which in turn are passed to the agent. The agent of course can be located in any machine. As the agent receives the event, the machine will inspect if there is any mirrors associated with that agent. If it has, it will also duplicate those events to the mirror agents.

In the event if the mouse cursor moves beyond the edge of a screen, the mouse agent (which corresponds to the mouse cursor) shall be migrated to another machine. In this case, only non-mirrored mouse agents are migrated. Any mouse agents marked as mirror should not be migrated and should be destroyed immediately.

Initially, GLIA employs relative coordinate (dx,dy) to steer the cursor. Thus if a mouse is moved 10 units to the upper left, (dx,dy) will be (-10, -10). However, this approach is unsuitable for mirroring due to coordinate mismatch between the server (source of cursor) and the mirror. The mirrors simply do not know

the initial coordinate of the mouse. Hence transmitting any (dx,dy) will result in the wrong coordinate. Certainly a solution to synchronize initial coordinate can be made, but there is a concern that it can incur overhead to the protocol since a synchronization mechanism needs to be introduced.

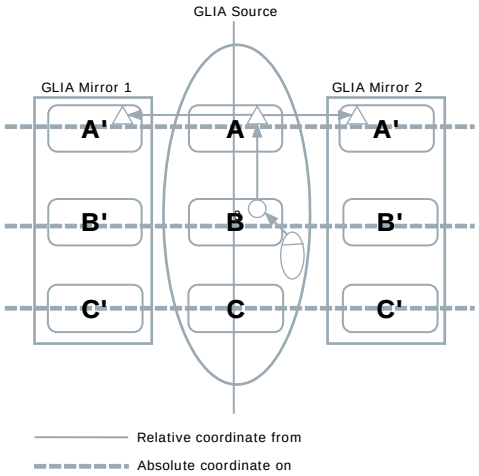

Figure 6. The extended GLIA cursor distribution system: There are 3 sets of GLIA environment. One is the source and the other two are mirrors. Cursor transmission within the same set of GLIA environment employs relative coordinate, while cursor transmission within different sets of GLIA environment employs absolute coordinate.

Consequently, a hybrid solution involving both coordinate systems is created. To enable smooth transition within screens of the same GLIA unit, relative coordinate is used. Then each machine will use the relative coordinate to acquire the absolute (x,y) coordinate. The result yielded is then transmitted to its mirrors. As illustrated in Figure 6, the physical mouse is located in host B. The mouse is managed by a monitor (circle), and the cursor is located at host A since the agent is there. The communication and transition from B to A employed the relative coordinate, as shown in the thick bold line. Then host A's agent will calculate the absolute (x,y) position on the screen and distributes them to the mirror agents at A', as shown in the thick dotted line.

Next, for the GLIA to support a true multi-user input in contemporary OS, an "artificial" input injection mechanism is developed. Many OS typically has a main message queue which dispatches OS messages to each application. Using the same message queue, we can inject control input messages directly into the individual application's message queue. This is illustrated in Figure 7.

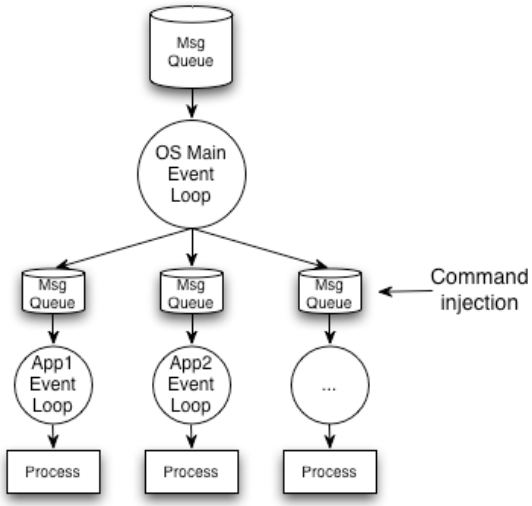

Figure 7. Typical operating system's application event distribution system.

5. DISCUSSIONS & RESULTS

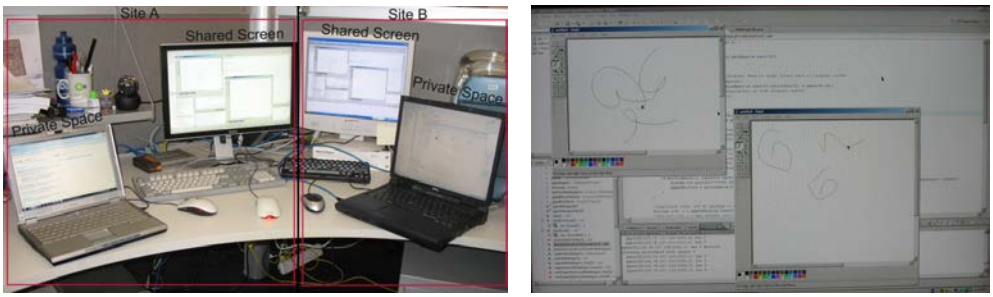

Figure 8. (Left) The M3DESKSHARE prototype in a private-public space configuration; (right) An ongoing M3DESKSHARE collaboration on the shared screen using two Windows Paint programs where two users input their mouse strokes onto both programs simultaneously without complication.

Figure 8 (left) shows the M3DESKSHARE prototype which demonstrates the MSS architecture. The configuration set up was a private-public screen sharing between two sites. These sites shared a single common screen labelled "Shared Screen". For each site, a private space was offered. The mouse cursors between the public and private space were linked. That implies the cursor can move from these spaces. As the cursor moved to the public space, it was reflected to the other site as well.

Figure 8 (right) shows the content of the "Shared Screen". There were a total of four cursors. Since each participant had their own control and M3DESKSHARE supports simultaneous multi-user input, users were able to work on separate applications simultaneously even though they belonged to the same desktop. In this example, two Paint programs were launched in Windows. If a time-slicing multi-cursor strategy was used, users would not be able to draw lines on the two paint programs simultaneously. Instead, the lines would have crisscrossed between the paint programs. With M3DESKSHARE, users can exploit desktop level multi-user interaction. That means two users can simultaneously work on their respective application. However, since the application itself does not support multi-user interactivity, users will not be able to enjoy application level multi-user support. If more than one user is controlling the same application, then the result will be unpredictable.

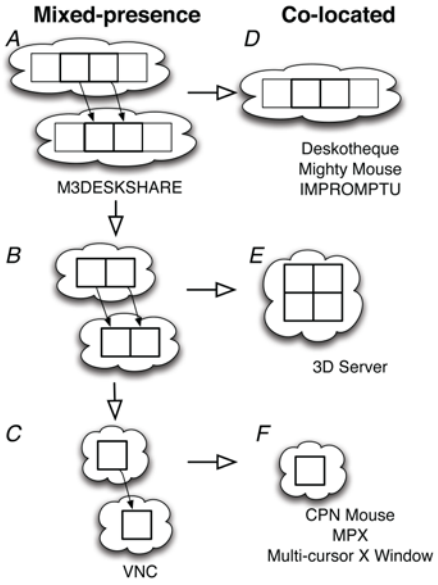

Figure 9. Derivatives of the configurations supported by the MSS architecture.

In principle, the MSS architecture supports screen sharing in a mixed-presence private-public configuration. We now show that from this configuration, it is possible to derive all possible configurations

arises from the above-mentioned prototypes, which belongs to one of the subset of the private-public configuration.

Figure 9 shows how the MSS architecture, represented by the M3DESKSHARE prototype, can be decomposed into various configurations. The left side of the figure shows collaboration performed over a network remotely. While the right side shows co-located collaboration, the machines are still linked over network in a local environment.

At the first level (A & D), the machines are configured as a public-private space. Figure D shows how multiple machines can be linked together for co-located collaboration. At the next level (B& E), the private space is discarded and all machines are exposed as public. So far, there is virtually no prototype that allows the sharing of multiple machines display remotely among multiple users. Finally, in its most primitive level (C & F), the MSS architecture can be utilized as a VNC-style desktop sharing such as the one illustrated in configuration C. If no remote machine is involved, then the configuration can be simplified to configuration F which is co-located collaboration. Under this configuration, only a single machine is involved and so only multi-cursor is needed in order to support multi-user collaboration.

6. CONCLUSION

In this paper, we have shown the capabilities required for a cursor system to support screen sharing in a multi-user multi-machine MDE in a mixed-presence environment. Apart from the stated capabilities, we have also introduced two primitive cursor functions:

- Cursor transition
- Cursor mirroring

Using these two primitive functions, we showed a cursor distribution system that can support MDE screen sharing in the public and private-public workspace. The system also ensures mouse cursor can navigate through a large tile display. The system is:

1. Cross-platform as it does not requires any modification to any part of the OS or windowing system.
2. Supports multiple machines.
3. Maximize multi-user support without time-slicing cursor.

This architecture does not intend to rival other architectures. Rather, we presented a case on why additional capabilities are required, and how different capabilities can be integrated. We also showed that the MSS architecture does not only support screen sharing in a private-public mixed presence environment, but it also supports screen sharing in any combination of mixed, distributed and co-located environment, single or multiple machines and private-public or public only configurations. This encompasses all scenario described to be supported by other prototypes as discussed previously.

REFERENCES

Biehl, J. T. *et al.* (2008) 'Impromptu: a new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development', in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 939–948.

Booth, K. S. *et al.* (2002) 'The mighty mouse multi-screen collaboration tool', in *Proceedings of the 15th annual ACM symposium on User interface software and technology*, pp. 209–212.

Davis, J. and Chen, X. (2000) 'LumiPoint: Multi-User Laser-Based Interaction on Large Tiled Displays', *Displays*, 23, p. 2002.

Gutwin, C. and Greenberg, S. (1998) 'Design for individuals, design for groups: tradeoffs between power and workspace awareness'. University of Calgary.

Hutterer, P. and Thomas, B. H. (2007) 'Groupware Support in the Windowing System', in *Proceedings of the Eight Australasian Conference on User Interface - Volume 64*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc. (AUIC '07), pp. 39–46. Available at: http://dl.acm.org/citation.cfm?id=1273714.1273721.

Jeong, B. *et al.* (2006) 'High-performance dynamic graphics streaming for scalable adaptive graphics environment', in *SC'06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, p. 24.

Myers, B. A. (1999) *An implementation architecture to support single-display groupware.*

Ng, C. J. and Takatsuka, M. (2009) ‘Method of Frame Buffer Transmission over Reliable Multicast Network’, in *CollabTech*. Sydney.

Robertson, G. *et al.* (2005) ‘The large-display user experience’, *IEEE computer graphics and applications*. IEEE, 25(4), pp. 44–51.

Sakurai, S. *et al.* (2008) ‘A middleware for seamless use of multiple displays’, in *International Workshop on Design, Specification, and Verification of Interactive Systems*, pp. 252–266.

Tang, A., Boyle, M. and Greenberg, S. (2004) ‘Display and presence disparity in Mixed Presence Groupware’, in *Proceedings of the fifth conference on Australasian user interface-Volume 28*, pp. 73–82.

Tse, E. *et al.* (2004) ‘Avoiding interference: how people use spatial separation and partitioning’, in *in SDG workspaces Proceedings of the 2004 ACM conference on Computer supported cooperative work, ACM*.

Westergaard, M. (2002) ‘Supporting Multiple Pointing Devices in Microsoft Windows’, in *Proceedings of Microsoft Summer Workshop for Faculty and PhDs*.

Yuizono, T., Nishimura, S. and Munemori, J. (2006) ‘Development of Middleware GLIA for Connective Wide Collaborative Space with Networked I/O Device’, in *Proceedings of the 10th International Conference on Knowledge-Based Intelligent Information and Engineering Systems - Volume Part III*. Berlin, Heidelberg: Springer-Verlag (KES’06), pp. 174–180. doi: 10.1007/11893011_22.