RemoteDT: Support for Multi-Site Table Collaboration

Alan Esenther, Kathy Ryall
Mitsubishi Electric Research Labs (MERL)
Cambridge MA, USA
{esenther, ryall}@merl.com

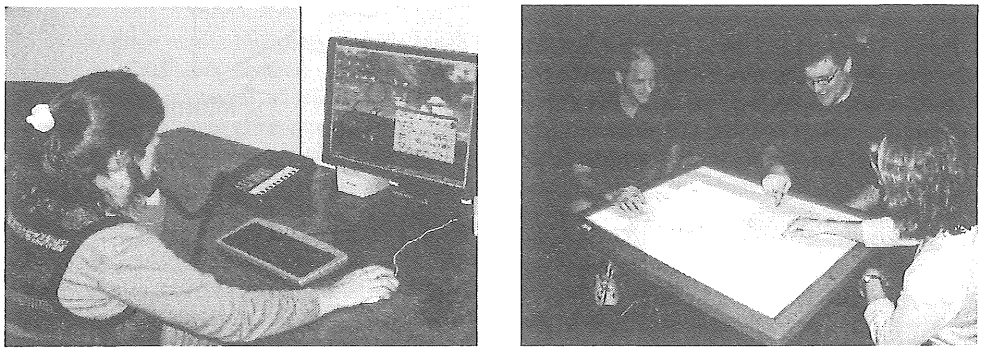

Figure 1. Example scenario for multi-site collaboration. (Left) Single user on her desktop system. (Right) Three people working around an interactive tabletop. Both sites see the same display. A detailed screenshot of the shared display is shown in Figure 4.

Abstract

Multi-user, interactive tabletops are an exciting new form factor to support collaborative group activities. To date, much of the research in this area has focused on co-located collaboration -- groups of people seated around the same physical table. A natural step is to connect a co-located group working around a table to other remote sites, creating a shared collaborative environment. In this paper we present RemoteDT, an initial prototype to explore the space of multi-site table collaboration. Our system was designed to exploit DiamondTouch [2], a multi-user multi-technology, but could utilize any table hardware that provides the same capabilities. RemoteDT provides three modes of interacting with the table (manipulation, annotation, and telepointers), as well as support for connecting to remote tables or single-user devices. We believe our initial exploration of the space will serve as a good starting point for future exploration by ourselves and others.

1. Introduction

Recent advances in touch technologies (e.g., DiamondTouch [2], SmartSkin[11], DViT[3]) have enabled research in new form factors for collaborative multi-user environments. Interactive, computationally augmented tabletops are one such area. To date, most of the research in this area has focused on co-located collaboration – groups of people seated around the same physical table. We ourselves have developed a number of concept prototypes to illustrate the capabilities of and explore the interactions on the tabletop form factor [4,6,9,12].

A natural next step is to connect an interactive table to remote locations to provide a collaborative environment for groups of people working together without being physically in the same physical space. While much of the early CSCW research specifically addressed this notion of remote or distributed collaboration, in most cases it assumed only one person at each remote location. Existing commercial remote conferencing solutions, such as the classic NetMeeting[9] application, require explicit turn-taking. Newer offerings such as Webex [13] and Bridgit [1] only let one person interact at a time at each location. All of these conventional conference solutions facilitate collaboration to a *group of distributed people*. In contrast, our goal is to support *distributed* ***groups*** *of people*.

Using a table for one or more of the remote sites allows a group of co-located people to work with groups or individuals at remote locations in a way that today's systems do not support. It also raises

new challenges in designing interfaces and interactions to exploit the capabilities of the multi-user tables.

As part of our ongoing research in shared-displays groupware we developed RemoteDT to begin to explore the space of table-to-table collaboration. Our system was designed to exploit DiamondTouch [2], a multi-user multi-touch technology that provides support for parallel concurrent input from multiple users and differentiates which touches comes from which users. Our system could, however, utilize any table technology that is able to provide the same capabilities. We used Commercial Off-The-Shelf (COTS) software, where available, for traditional remote collaboration needs, and focused our work on the table-specific aspects of the collaboration.

In the next section we describe a sample multi-site collaboration usage scenario that we used as a case study to design and implement RemoteDT. In Section 3 we present its three interaction modes (mouse mode, annotation, and custom pointers), followed by the system implementation details. In section 4 we summarize the lessons learned and directions for future work.

2. Usage Scenario

Our usage scenario was based on a field engineering task. In particular, we developed a system to support remote field engineering, in which a field engineer is dispatched to a particular site to do maintenance or repair on some equipment. In some cases the engineer may be able to complete his job on his own. In other cases he requires additional information and guidance from remote collaborators. In our example, two groups of different people are called upon to evaluate the information gathered by the field engineer, identify the problem at hand, and construct a plan of action to solve the problem. The two groups are located at different sites, which are distinct from the location of the field engineer. Figure 2 depicts our usage scenario.

Each site plays a key role in the overall collaboration. The first group, the manufacturers (top left Figure 2), has detailed knowledge of the machinery in need of repair, including wiring diagrams and mechanical schematics. The second group, the plant operators (top right Figure 2), has details of their particular installation along with audit trails and sensor readings of the equipment in question. Finally, the field engineer is on-site with the equipment itself, and can provide real-time information about the equipment. He also has use of his cell-phone to send and receive images of any relevant components or environmental issues. Together the manufacturers and the plant operators must coordinate the activities of the field engineer.

While our particular distributed collaboration scenario involves two groups of people working together, each with an interactive tabletop system at their disposal, our system supports traditional single-user devices (e.g., desktop or laptop) at sites as well. Likewise, other mobile devices could be used in place of our field engineer's cell phone.

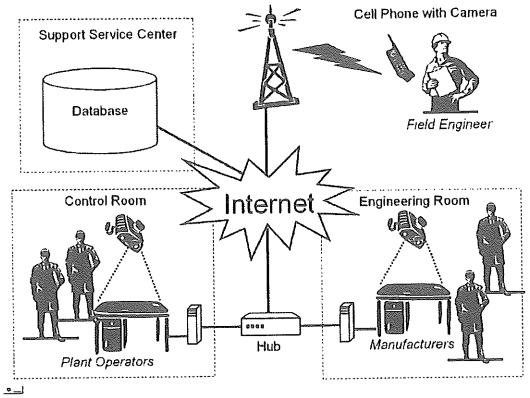

Figure 2. Our sample usage scenario connects two tables at remote sites with a field engineer located in a third location. The tables share the same virtual display while the field engineer interacts over cell-phone based email and voice.

We believe our sample scenario is typical of many distributed collaborative activities in that multiple parties are working together on a common task, each party has particular information that may not be accessible or apparent to the other parties, and one of the parties involved may itself be a group of co-located people working together. Other examples of this type of activity include command and control centers, emergency response situations, and urban planning activities to name a few.

3. Interaction Modes and Implementation

In many distributed collaborative settings each group is likely to be using its own (potentially proprietary) tools. Thus one of our goals was to allow each site to use its own software applications – in essence arbitrary legacy applications.

While we hope that in the future multi-user applications will be readily available, today's applications are primarily single-user in nature. While it is possible to augment some existing applications through a programmatic interface (e.g., Microsoft's COM-based interfaces to control Microsoft Office applications), it is impossible to do so in all cases. In our efforts to support the use of legacy applications in collaborative table settings, we noted the need for three distinct interaction modes to

support table-interactions when connected to a remote site.

When using a direct-touch input device, every touch counts – the underlying software must interpret it in some way. But the question is…did someone intend to interact with an application running on the table? Or did they merely wish to indicate or reference content on the table (i.e., a deictic reference). And in the case of remote collaboration, sometimes you need to visually highlight some particular region of content to make it clear – to draw someone else's attention to something.

Each of our three modes interprets a user's touch on the table in a different manner. Users manage each mode through a control panel – a persistent window that "floats" above all other applications (see Figures 4-5). It was designed to emphasize the three interaction modes, while letting users see at a glance what tools and capabilities are available in each mode. The control panel can be minimized leaving a single push-button (which is always visible and accessible) to reopen the control panel.

3.1. Mouse Mode

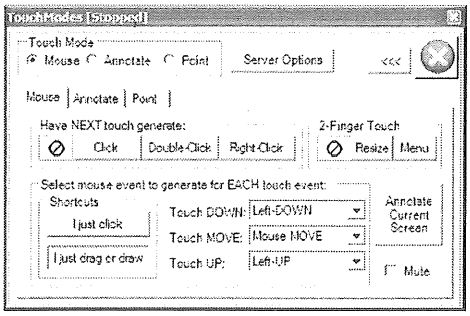

Figure 3. The Mouse Mode control panel provides an easy mechanism to map touches to mouse functions.

Our first mode of interaction is mouse mode; touches to the table are intended to manipulate an application. It enables users to interact with arbitrary applications via mouse emulation. RemoteDT was one of our earliest explorations into mouse emulation. In this system we provide access to different mouse functionality through an easy-to-use control panel (see Figure 3).

Shortcut buttons determine whether touching the table just moves the mouse (i.e., providing mouse over support) or generates a left button down (i.e., drag or draw events). Users can arm the next touch with different button activations (click, double-click, right-click). While touching with a single-finger controls mouse position, two-finger touches are mapped to other functionality such as resizing a window or opening a context menu.

An interesting aspect to our mouse emulation is that while the table itself is able to provide multi-user support, in particular concurrent input from multiple people, today's mouse is inherently single-user. In earlier versions of our system we relied on social protocols to coordinate and serialize user interactions in mouse mode. We quickly realized that people did not always wait to "grab" the mouse until someone else was finished. As a result we decided to implement a first-come first-served mouse policy. The first person to touch the table gains control of the mouse. While that person is interacting with the table any touches by other users is ignored.

3.2. Annotation

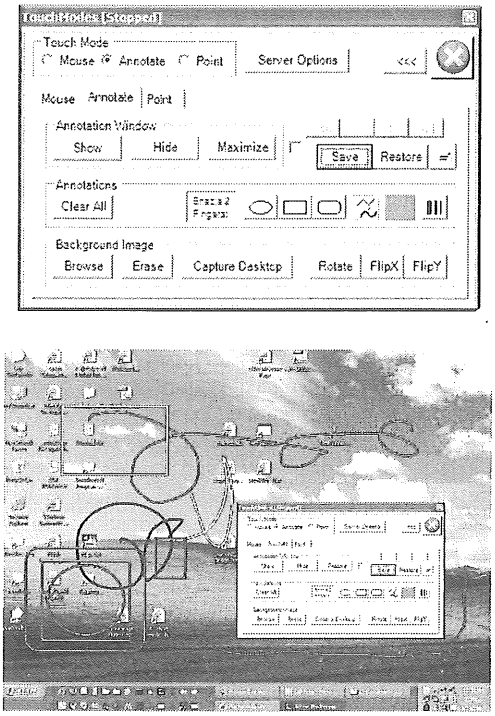

Figure 4. (Top) Annotation Control Panel Detail: The Annotation control panel allows users to customize their "ink" as well as choose the image to annotate. (Bottom) Sample screenshot of the RemoteDT session from Figure 1: Users have annotated this screenshot, each in a unique color. The yellow border indicates a static image rather than a live desktop.

Our second mode of interaction is annotation; touches to the table are intended as ink. It enables users to mark up arbitrary screen contents as if they

were writing with a pen. A control panel (Figure 4) enables each user to select their annotation color and adjust other typical drawing parameters (e.g., line smoothing and width). Touching with a single finger draws a line. Touching with two-fingers creates a user-specific shape (e.g., circle, rectangle, or rounded rectangle). Sample annotations are shown in Figure 4(bottom).

Ideally our annotation mode would work directly with underlying applications, embedding markup directly into the documents themselves. In practice this would require very large engineering efforts and, in cases of proprietary software, is impractical. To solve this problem RemoteDT provides a "Capture Desktop" button in annotate mode to grab a copy of the current desktop image. The captured desktop image will not include the Control Panel itself. The mouse mode also has a shortcut button called "Annotate Current Screen" (see Figure 3) which captures the current desktop image (sans Control Panel) and switches to annotation mode.

In annotation mode users may load any arbitrary image file to use as the background image on which to annotate. To support users working at arbitrary positions around the table, the contents can be rotated by 90 degrees or flipped along either its x or y axes.

RemoteDT also provides special support which allows the field engineer to email an image to the RemoteDT application, which immediately switches to annotation mode and displays the image, ready for discussion and annotations. A copy of the image currently being annotated can also be emailed, complete with annotations, to any email address. This allows participants to send reminder copies of the discussion to themselves, and also enables parties not currently using the RemoteDT system to have access to some of the session information. In our example scenario these features were used to exchange information with the field engineer.

3.3. Custom Pointers

Our third mode of interaction is custom pointers; touches to the table are intended to manipulate floating pointers which create a tele-presence. It enables users to position their "pointer" anywhere on the display as a means of referencing particular content. Each telepointer [7] is a color-coded text label (i.e., user's name or title, say) and can be overlaid on any application. The Pointer control panel (Figure 5) allows users to customize their pointer, including deleting or hiding it as they desire.

Figure 5 shows the Pointer control panel and includes five telepointers – four from people co-located on a tabletop and one from a remote laptop user. Each pointer also includes an arrow, which is user-configurable to point to the left or right; touchers "0 through 3" are pointing to the left, while the "custom label" points to the right.

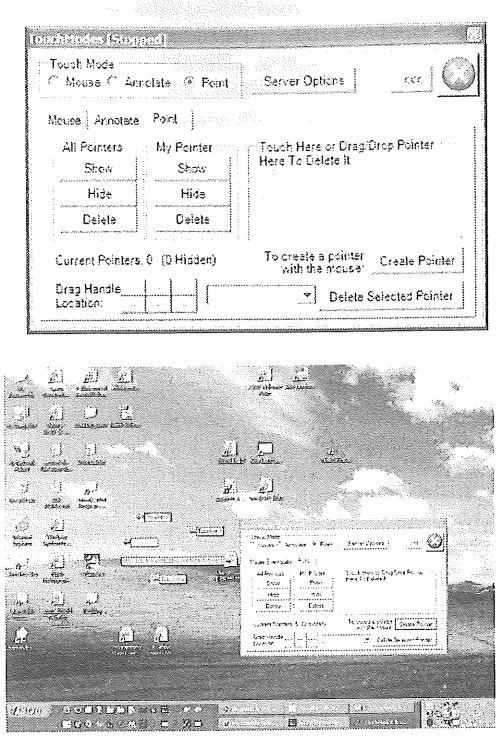

Figure 5. (Top) Pointer Control Panel detail: The Pointer control panel allows users to customize their telepointers. (Bottom) A "live" screenshot of RemoteDT in action: Five telepointers are shown on the background image. One telepointer with a custom label points to the right.

To support users seated at arbitrary positions around the table, each pointer also has a custom handle position which can be adjusted on a per-user basis through the control panel. The "drag handle location" widgets (lower left of Figure 5(top)) are an instance of iDwidgets [12]; each user's identity is passed as a parameter to the widget so that the widget "knows" whose settings to change when touched. The "drag handle location" refers to the point on an object which will track to a person's touch on the table. For a mouse pointer this corresponds to the "hot spot" of the cursor. When using a direct-touch input, however, it is often desirable to shift the position of the "handle" based on a person's position (e.g., around a table) and possibly which hand he or she is using to prevent the person's hand from occluding the object (or content) being manipulated.

3.4. Implementation

RemoteDT was developed on the DiamondTouch platform [1,2]. DiamondTouch is a multi-user, multi-touch technology that supports simultaneous input from up to four users, and differentiates who is touching where. It also provides multi-touch interaction, enabling each user to interact with one finger, two fingers, one hand, two hands and so on. Each DiamondTouch table is front-projected to create coincident input and output. RemoteDT was implemented in .NET and used the .NET Remoting framework to distribute touch events and some state information.

Our goal was to explore the table-specific components needed to support multi-site collaboration – the "tableness" of the collaboration rather than the underlying enabling technologies. RemoteDT relies on existing Commercial Off-The-Shelf (COTS) software to provide screen-sharing functionality. This allows us to leverage existing work and quickly adapt new offerings without getting bogged down in an area of development to which others are already dedicating substantial resources. Similarly, we assume that audio is shared via simple speakerphones.

As was mentioned earlier, to support the larger scenario described in Section 2, RemoteDT provides on-screen notification of incoming email, as well as methods for sending email to a remote participant. To provide backwards compatibility for non-table sites, RemoteDT supports regular mouse input, and allows a physical mouse to be mapped to an arbitrary user via the keyboard. Thus RemoteDT runs without any DiamondTouch tables at all, although to support multiple users in one location (i.e., on one CPU) a multi-user technology such as DiamondTouch is needed.

Figure 1 shows a two-site collaboration conducted using RemoteDT. On the right a co-located group is seated around an interactive table. On the left we see a single-user with a traditional desktop setup. The two sites share the same virtual display – its screenshot is shown in Figure 4. Site A uses a speakerphone connection while site B uses a telephone.

In practice RemoteDT supports multiple remote tables, as illustrated in Figure 2. Conceptually it can support arbitrary numbers of tables and remote users, but for our initial implementation we set a limit of up to ten simultaneous users for the system. Larger group collaboration, whether with co-located or distributed groups, has its own set of challenges, including issues of surface size and visual clutter.

4. Discussion and Future Work

While we developed RemoteDT with a concrete usage scenario in mind, we have used the system to begin to explore the space of multi-site table collaboration. Through our design and implementation we have learned a number of lessons that we believe will be helpful to others working in the areas of table-based collaboration.

The first set of issues surrounds the need and use of mouse emulation. First and foremost, for any new technology to be accepted, it must provide backwards compatibility for legacy software and applications. Concretely, while we hope that tomorrow's table-based applications break away from today's WIMP tradition, we realize that to be useful today tables must provide mouse emulation.

The control-panel based interactions provided by RemoteDT were our first exploration into mouse emulation for direct-touch table input. More complete gestural interfaces can also be used to provide mouse emulation without the need to display an on-screen control panel. In [5] we describe a fluid mouse emulation approach; we also provide a more thorough discussion of the challenges associated with trying to provide the functionality of a multi-button mouse with a direct-touch input device. In addition, we have found mouse emulation to be such a useful utility that we have broken it out as a stand alone utility and have migrated it to the system tray; it has become a standard, and perhaps the most-often used, component in the DiamondTouch SDK [4].

Also of note, today's mouse and applications (with the notable exception of MID [8]) are inherently single-user in nature. While we enforce serial mouse use, other schemes are possible for coordinating and resolving contention for the mouse (e.g., [9]), but are beyond the scope of this paper. The notion of a multi-user mouse, or conversely how to support multiple mice (and in turn multiple foci), raises a number of interesting questions. Our future work will explore this area and we encourage others to do so as well.

Another important issue came to light with respect to annotations. When using annotation mode, people often became confused about the state of RemoteDT and tried to manipulate widgets and content in the static image as if they were interacting with the live desktop. To disambiguate the static image from the live desktop we added a border to the image displayed as the background in annotation mode (see Figure 4). In contrast, in Figure 5 (bottom) we see a "live" desktop; it has no border.

Future work includes investigation into annotating on a live desktop image. This feature would allow people to either annotate or interact with visible content rather than having to explicitly

switch between modes as in the current RemoteDT implementation. Likewise, switching between our current three modes more seamlessly is also left for future work.

Finally we should note two issues that we explicitly chose not to explore in RemoteDT. First RemoteDT makes only limited use of the "around-the-table" nature of a remote table. An obvious next step would be to integrate some of the features from RemoteDT with a tabletop toolkit such as DiamondSpin [13]. At the time of our work, however, DiamondSpin was limited in its support of legacy applications. Second, RemoteDT's use of telepointers only scratches the surface on the issue of telepresence. Tang et. al. [14] explore the use of video capture in Mixed Presence Groupware. Integrating RemoteDT into such a system would also be another interesting avenue to explore.

5. Conclusion

We have presented RemoteDT, an initial prototype to explore the space of multi-site table collaboration. RemoteDT facilitates simultaneous interaction between distributed groups of people (as opposed to a group of distributed people) where each person potentially has multiple points of interaction. It provides three modes of interaction via control-panel access, letting users see at a glance what tools and capabilities are available in each mode. All of today's commercial conferencing solutions tend to require turn-taking, and fall short of allowing the full breadth of multi-touch and multi-user interaction afforded by RemoteDT.

RemoteDT was developed for a distributed collaboration task centered on field engineering. Such tasks are common across a wide number of domains; we hope that our initial exploration of multi-site table-based collaboration will aid others as they begin to explore the area.

5. Acknowledgements

The authors wish to thank their collaborators at Mitsubishi Electric Corporation in Japan for providing motivation and domain expertise on this project. We are especially grateful to Mr. Iku Ikeda and Mr. Mitsuru Nakadan.

6. References

[1] Bridgit (www2.smarttech.com).

[2] Dietz, P., and Leigh, D. "DiamondTouch: A Multi-User Touch Technology," *ACM Symposium on User Interface Software and Technology (UIST)*, pp. 219-226, November 2001.

[3] DViT (www.smarttech.com/DViT/).

[4] Esenther, A.; Forlines, C.; Ryall, K.; Shipman, S., "DiamondTouch SDK: Support for Multi-User, Multi-Touch Applications," *Demonstration at ACM Conference on Computer Supported Cooperative Work (CSCW)*, November 2002. Available as MERL TR2002-048.

[5] Esenther, A.; Ryall, K. "Fluid DTMouse: Better Mouse Support for Touch-Based Interactions," To appear in Proceedings of Advanced Visual Interfaces (AVI) 2006.

[6] Esenther, A.; Wittenburg, K., "Multi-User Multi-Touch Games on DiamondTouch with the DTFlash Toolkit," *Intelligent Technologies for Interactive Entertainment (INTETAIN)*, November 2005.

[7] Greenberg, S., Gutwin, C. and Roseman, M,. "Semantic telepointers for groupware," In *Proceedings of the OzCHI '96 Sixth Australian Conference on ComputerHuman Interaction,* Hamilton, New Zealand, November 1996.

[8] Hourcade, J.P and Bederson, B.B "Architecture and Implementation of a Java Package for Multiple Input Devices (MID)," HCIL-99-08 , CS-TR-4018 , UMIACS-TR-99-26, May 1999.

[9] Morris, M. R..; Ryall, K.; Shen, C.; Forlines, C.; Vernier, F., "Beyond Social Protocols: Multi-User Coordination Policies for Co-located Groupware," *ACM Conference on Computer Supported Cooperative Work (CSCW)*, November 2004.

[10] NetMeeting (www.microsoft.com).

[11] Rekimoto, J. "SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces," CHI2002.

[12] Ryall, K.; Esenther, A.; Everitt, K.; Forlines, C.; Ringel Morris, M.; Shen, C.; Shipman, S.; Vernier, F., "iDwidgets: Parameterizing Widgets by User Identity," *IFIP TC13 International Conference on Human-Computer Interaction (Interact)*, September 2005.

[13] Shen, C.; Vernier, F.D.; Forlines, C.; Ringel, M., "DiamondSpin: An Extensible Toolkit for Around-the-Table Interaction," *ACM Conference on Human Factors in Computing Systems (CHI)*, pp. 167-174, April 2004.

[14] Tang, A., Neustaedter, C., and Greenberg, S. "Embodiments and Video Arms in Mixed Presence Groupware," Report 2004-741-06, Department of Computer Science, University of Calgary, March 2004.

[15] Webex: (www.Webex.com).