Spilling: Expanding Hand held Interaction to Touch Table Displays

Dan R. Olsen Jr., Jeffrey Clement, Aaron Pace
Computer Science Department, Brigham Young University
olsen@cs.byu.edu

Abstract

We envision a nomadic model of interaction where the personal computer fits in your pocket. Such a computer is extremely limited in screen space. A technique is described for "spilling" the display of a hand held computer onto a much larger table top display surface. Because our model of nomadic computing frequently involves the use of untrusted display services we restrict interactive input to the hand held. Navigation techniques such as scrolling or turning the display can be expressed through the table top. The orientation and position of the hand held on the table top is detected using three conductive feet that appear to the touch table like three finger touches. An algorithm is given for detecting the three touch positions from the table's sensing mechanism.

1. Introduction

There is a great attraction to carrying a personal computer in your pocket. It enables a nomadic style of computing that is not feasible with desktop machines or even with laptops. There are now many hand held devices that are less than one cubic inch in volume and yet have much more storage and processor power than the original Macintosh. However, the user interface to such devices is unacceptable for many applications because of display size.

The XICE project (eXtending Interactive Computing Everywhere) seeks to address the problem of very small personal computers by annexing screens where we find them in the world. XICE shares many of the goals of the Personal Server [13, 7] project as well as our own Join and Capture [6] system. Our approach stands in contrast to Pebbles [5] which uses hand held devices as controllers for applications running on other machines. In the XICE world the hand held computer is the personal computer and other devices are simply interaction servers. The advantage of this approach is that compatibility issues are sharply diminished. X-Windows [10], Virtual Network Computing (VNC) [8] and the web have all shown the compatibility advantages of standardizing interaction services.

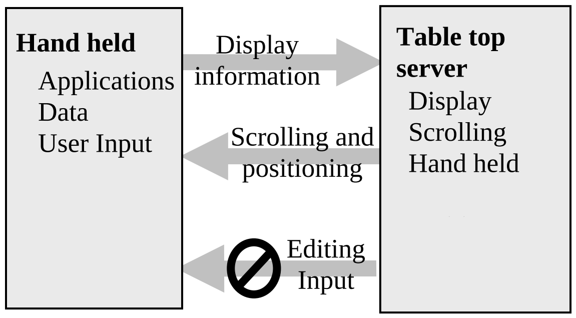

Fig 1. Hand held / Table top relationship

There are two specific challenges in this vision that we will address in this paper. The first is the integration of the hand held computer with the display services that it has annexed. We will show how the display of a hand held can be "spilled" onto a table top display to create a Focus-Plus-Context [1] expansion of any hand held application. The second issue is the security of the hand held computer. In a nomadic model of computing that uses devices encountered "in the wild" it is not safe to assume that such devices can be trusted. In this work we resolve this problem by performing most interaction locally within the hand held personal computer. We only accept navigation inputs, such as scrolling, from foreign devices. Accepting full user input from a foreign device essentially cedes control to that device with all of the accompanying dangers. Figure 1 shows the relative relationships between hand held personal computers and table top servers.

Figure 2 shows our Spilling prototype with the hand held laid on the surface of a Diamond Touch table. In this figure the application is a spreadsheet that the user is scrolling by dragging their finger across the surface. The entire application is displayed on the table surface with the interactive focus of the application displayed on the hand held. In terms of usability we compare Spilling to what is possible on a hand held alone as opposed to what might be possible on a desktop computer. The table top context of Spilling greatly enhances the ability of a user to interact with the hand held.

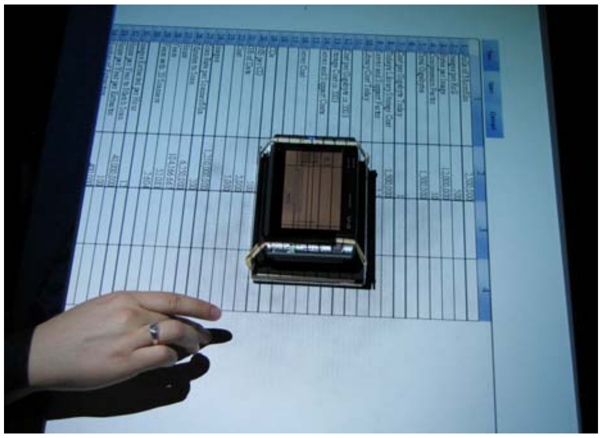

Fig 2. Hand held on table top display

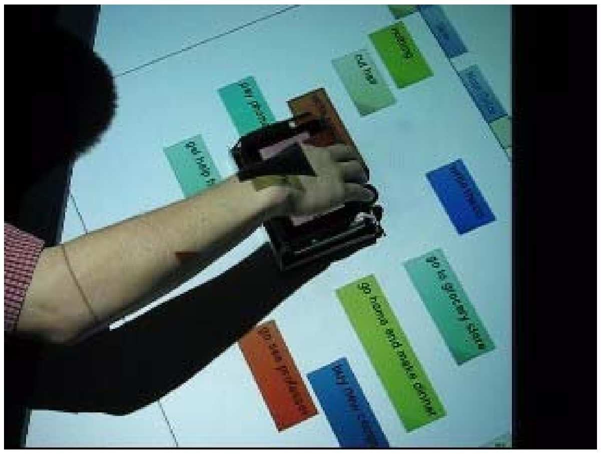

Fig 3. Repositioning hand held

Spilling draws its inspiration from Focus-Plus-Context displays [1]. The focus is on the hand held where the user can safely interact using the same interactive techniques normally used on the hand held. The context is provided by the table top display with a touch sensitive surface providing a mechanism to scroll information into the focus region. The idea is that this is a hand held computer that behaves as it always did, but it has been supplemented by this large table top display. Unlike Focus-Plus-Context which has a rather rigid configuration, the user can move the hand held around the surface and position it in whatever place is comfortable to work. Figure 3 shows the user rotating the hand held to work more comfortably on a “sticky note” application for organizing ideas. This would be similar to tipping a sheet of paper in order to write more comfortably.

When the hand held moves, the work moves with it. When the user scrolls with their finger the work moves beneath the hand held. This is different from Ubiquitous Graphics [9] where the wall display is fixed with the smaller device moving relative to it. This fixed-world approach is also found in Peepholes [15] where the hand held device is moved relative to a large fixed display that is only shown through the hand held device providing no peripheral context. The problem with fixed-world display models is that the user can be forced to work in awkward physical positions, such as the extreme top of the display. In Spilling the user can move the work to where it is most comfortable regardless of the physical size of the contextual display. This hand held centric approach also allows the user to rotate the hand held so that the display can be seen by someone across the table as in DiamondSpin [11].

In the remainder of this paper we will address Spilling’s architectural and security issues. First issue is distributing the display from the hand held to the table top. This is mostly handled by the XICE architecture. The second issue is the geometry synchronization between the hand held and the table top. Third there is the problem of sensing the location and orientation of the hand held personal computer. Lastly the UI security threats are addressed.

2. XICE Architecture

The full XICE architecture is beyond the scope of this paper. What are presented here are the essentials for implementing Spilling. The goal of XICE is to explore the software architectures that can make highly nomadic computing possible without sacrificing display or interactive capabilities. Because we wanted to rethink mobile interaction architectures we have abandoned any support for legacy software. We feel that nomadic computing is compelling enough to warrant rewriting applications to a new platform. Using XICE we have implemented a text editor, a “sticky notes” idea organizer, a presentation tool (like PowerPoint), a drawing application, and a spreadsheet as well as the games of checkers, tic-tac-toe and Risk. These give us a rich application base for exploring architecture ideas.

XICE applications are organized around *sheets*, which are similar to traditional windows. A sheet is either *local* or *remote*. A local sheet is drawn on the screen of the same computer where the application is running. A remote sheet is drawn on some machine that is accessed over the Internet.

Associated with each sheet is a *presentation tree* that represents everything to be drawn on that sheet. These presentation trees are like the scene graphs in 3D systems and the drawing architecture of Piccolo [2]. Rather than implementing the damage/redraw technique of most interactive graphics, XICE applications interact by modifying the presentation tree. Tree modifications propagate up the tree to the sheet. If the sheet is local then XICE handles the necessary damage/redraw activity to get the screen redrawn. If the sheet is remote then presentation tree changes are serialized to the remote machine where they are used to update a copy of the presentation tree. That copy is then redrawn on the remote display In Spilling all Internet access is via a wireless 802.11g connection from the hand held. The change propagation and serialization algorithms are beyond the scope of this paper. The key feature is that modifications to a transformation require just a few bytes of network traffic to modify the corresponding remote transformation node rather than the traffic for the complete redraw required in X or VNC. Since most of Spilling is transformation modification this makes the whole technique very network-friendly. The distribution of presentation trees is hidden from and irrelevant to XICE applications.

There are two points in the XICE drawing model that are key to the Spilling implementation. As in 3D scene graphs some interior nodes in a presentation tree can be geometric transformations (scale, rotate, translate or any combination). A transformation node modifies the way in which all of its children are drawn. A second feature is that all drawing is defined in View Independent Coordinates (VICs). VICs are defined to deal both with differing screen resolutions and with differing viewing distances. The informal definition is that readable text should be at least 10 VICs high. The scaling of VICs to actual pixels is handled deep in XICE where the actual display configuration is known. For each display XICE is configured with the parameters *pixels per inch* and *viewing distance* in inches. From this the VICs-to-pixels scaling is computed. This will be important in coordinating table geometry with hand held geometry.

Figure 4 shows how Spilling is implemented in the XICE architecture. At the heart of the implementation is the application's presentation tree. This embodies everything that is to be drawn in the application as well as its input event processing. Directly above the application's tree is the scrolling transformation node **S**. This node positions the application work in response to the user's dragging with their fingers. Above the **S** node is a special *multi-parent* node that is responsible for synchronizing the hand held display with the table top display.

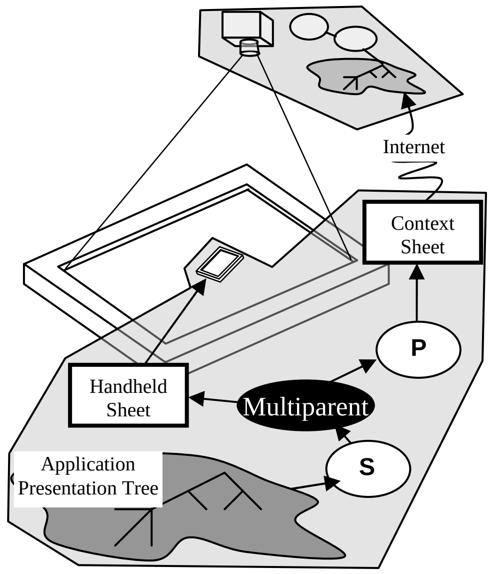

Fig 4: Spilling's XICE implementation

Also above the multi-parent node is the transformation node **P** that manages the position and orientation of the hand held. The node **P** is connected to the remote sheet for the table top context. This remote context sheet synchronizes over the Internet with the table top display's version of the presentation tree. The entire application is presented to the hand held, where most is clipped away, and also to the table top, where a more global view of the application appears. As figure 4 shows, almost everything about the application and its relationship to Spilling is contained in the hand held. The only part that is controlled by the foreign table top server is a copy of the presentation tree and the scrolling/positioning inputs.

When a change is made to the presentation tree by the application it is propagated up the tree to the multi-parent node. The multi-parent node sends the change notification both to the hand held's sheet and to node **P** where it is propagated to the context sheet and serialized to the table top display service. Both the hand held and the table top have presentation trees that reflect the current visual state of the application.

When the hand held needs to redraw its sheet for whatever reason it recursively traverses the tree down through the multi-parent node, bypassing node **P** but passing through node **S**. The position of the hand held on the table top has no effect on what should be drawn on the hand held. When the context-sheet is serializing it does pass through transformation **P** so that the hand held's position is taken into account when drawing on the table top.

The multi-parent node makes the whole display structure technically a directed-acyclic graph. However, the multi-parent node localizes the non-tree behavior that is required to get different displays drawn in different places without the application knowing about the duplication. The application is oblivious to the existence of the Spilling implementation above it in the tree. When the user interacts with the application itself through the hand held, all input events are distributed downward and transformed by $\mathbf{S}^{-1}$ before being sent to the application for processing. The scrolling of the display performed by Spilling is thus invisible to the application.

3. Table top interaction

When the user moves the hand held, as shown in figures 3 and 5, its location and orientation must be sensed by the table top server. This location and orientation is sent back across the Internet to the context sheet and transformation node **P** is modified appropriately as shown in figure 5. This change to **P** is serialized and returned to the table top display server which causes the table top display to update.

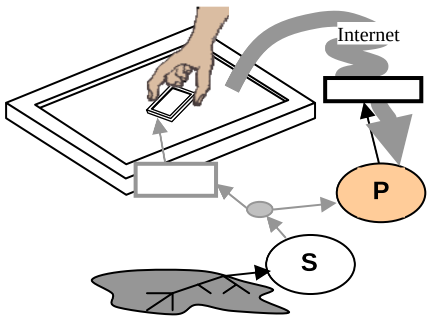

Fig 5. Changing hand held position

When the user scrolls the work under the hand held as shown in figure 2 a translation is sensed by the table top server. The change between the start and end points of the scroll is sent to the hand held and is propagated from the context sheet down through $\mathbf{P}^{-1}$ and used to modify transformation **S** as shown in figure 6. This change to transformation **S** is below the multi-parent node. Thus its modification is propagated to both the hand held sheet for redrawing and to the context sheet for serialization to the table top. XICE's parsimonious network usage makes these changes much more efficient than VNC or X. In neither of these table top interactions has the presentation of the application been sent over the network. That would only happen if the application itself changed its presentation tree and then only the relevant changes would be sent. This allows spilling to move both displays at interactive speeds.

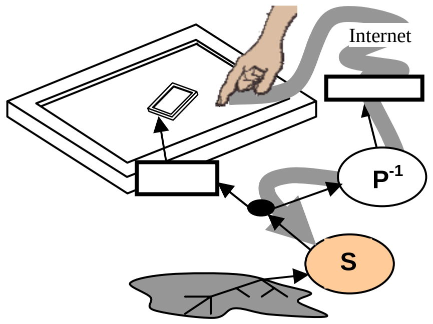

Fig 6. Finger scrolling

4. Table top technology

The table top server needs to support three tasks: display, scroll sensing and hand held position sensing. Display can either come from below or above. Display from below has many advantages in terms of installation, occlusion and elimination of shadows. Spilling is largely agnostic about the display choice except that dragging a hand held across a transparent screen may lead to excessive scratching. Projection from above would not have the scratching problem, but does create shadows.

Users might express scrolling using either a finger or a stylus. For a stylus-based technique a tablet is perfectly adequate. However, for scrolling we do not need the precision of a stylus and we would prefer not to incur the usability burden of finding and manipulating a stylus. Fingers rarely get lost.

The key sensing requirement is the location and orientation of the hand held computer. For this there are several competing technologies: cameras, structured light, pen tablets and the Diamond Touch. It would be possible to place fiducial marks on the back of the hand held computer so that a camera from below could sense the location and position. Two unique marks would clearly identify the location and orientation of the device. Camera from below poses the same surface wear problems as projection from below. However, fiducial marks can readily be painted or stuck on the bottom of the hand held to provide the necessary two points. PlayAnywhere [14] describes a set of such fiducial marks. Camera-from-below is a strong candidate for Spilling.

It is possible to place fiducial marks on the top of the hand held and put a camera above to sense the necessary three points. PlayAnywhere demonstrated that an oblique camera angle can be effective with little interference. However a quick glance at figure 3 shows that most fiducial marks would be obscured by the user's hand at exactly the time when we want to sense the hand held position. Camera from above would not be a good choice.

A structured light solution is also possible such as that proposed by Lee, et. al. [4]. This would require light sensors at the corners of the hand held device that could sense the position from the projected light signature. There are several problems here. The sensors must be manufactured into the hand held device which would make it more expensive and require a lot of cooperation between table top and hand held manufacturers. Structured light would interfere with the table top projection unless infra-red was used. Lastly the hand would obscure the sensors just as with top-side fiducial marks.

Ubiquitous Graphics [9] uses acoustic pen technology to sense the location of the focus display. Again this requires the manufacturers of hand held devices to include special hardware. The ultrasonic emitters are also large enough to add significant bulk to the hand held.

It would also be possible to use Wacom [12] tablet technology for sensing location and orientation. Embedding Wacom stylus hardware in the hand held would provide the necessary input. The passive stylus hardware could also be constructed as a "stick on" that could be attached to any hand held. This would provide an effective and accurate mechanism that would work on any of Wacom's tablet displays.

Our Spilling prototype uses the Diamond Touch table [3] as its sensing device. It has a wear resistant surface coupled with very simple user devices (fingers). Scrolling is easy by sensing the movement of a finger on the touch surface. The sensing of the hand held is more problematic. The Diamond Touch functions by sensing capacitance changes from users sitting on a conductive pad. If the chair pad causes installation difficulties we have placed it at the edge of the Diamond Touch table where it can be touched with the non-dominant hand.

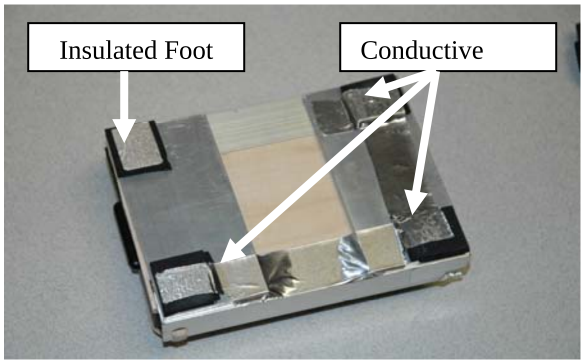

Fig 7. VAIO prototype with conductive feet

In figure 7 we show a conductive frame which we have attached to a Sony VAIO-UX hand held computer. When the user grasps the frame it becomes part of the Diamond Touch sensing circuit for that user. The result is that the three conductive feet appear to the Diamond Touch as if three of the user's fingers were touching the surface. In a real product the hand held's case would be conductive with one nonconductive and three conductive feet on the back. This would be inexpensive to manufacture into a hand held device. The VAIO is larger than we envision for most hand helds, but it was much simpler to program for this prototype.

5. Detecting hand held position/orientation

The remaining challenge is to locate the three conductive feet from the VAIO and convert their locations into the translation and rotation transformation **P**. The Diamond Touch is a projective sensing device. The locations of the touches are not sensed directly, rather a histogram of touch strength is produced for the individual X and Y coordinates. For a single touch the high points in the X and Y histograms are detected and the touch location is known. Projective sensing is not unique to the Diamond Touch. Most tablets are projection sensed through vertical and horizontal wires. Many IR-based optical tablets are also projection sensed. Projection sensing is economical because its cost grows with the perimeter length rather than the area of a sensing surface.

Projective sensing works great for single points, but it leads to ambiguity when sensing multiple points. For sensing the hand held's size, location and orientation we need to sense three unique points. However, the X, Y projections of those points are not unique. We resolve this using three techniques: 1) a known starting orientation, 2) the rigid configuration of the hand held's three conductive points and 3) continuity of motion.

Figure 8 shows the 4 possible sensing cases that must be considered in resolving the X, Y projections into the positions of 3 points on the handheld. Each case is identified by the number of peaks in each of X and Y. There are several variations of each of these cases. The numbered points for Case 1 indicate the identity of the three conductive feet on the bottom of the handheld device.

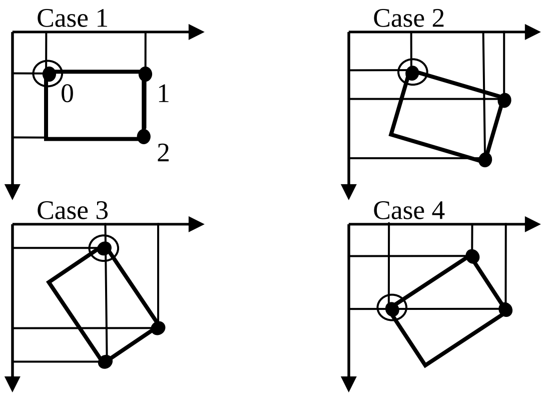

Fig 8. Cases for projection of 3 points

There is underlying software that examines the histograms from the Diamond Touch hardware and identifies the peaks that correspond to touches on the surface of the table. This histogram analysis layer returns the positions of 1, 2 or 3 peaks in each of X and Y. For the handheld device there are always 2 or 3 peaks in each of X and Y.

5.1. Case 1

Case 1 shows as 2 peaks in X and 2 in Y. Case 1 is the most ambiguous alignment and also the position for the hand held when the system is initialized. In the initial position the locations of feet 0, 1 and 2 are easily derived from the histogram information. From this position the width and height of the handheld can also be measured. It is essential that the device not be square. Having a distinct difference between width and height is critical in resolving the point positions in the other cases.

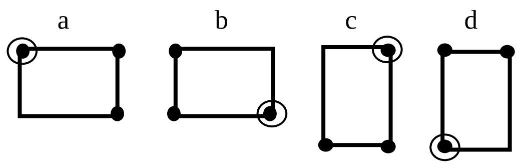

Fig 9. Variations on Case 1

Case 1 not only applies to the initial position, but also four other variations as shown in figure 9. Each of these shows as 2 peaks in each of X and Y and might occur in the course of the user moving the handheld around. Variations a and b can be distinguished from c and d by the known width and height of the handheld. Further ambiguity (a vs. b, or c vs. d) can be resolved by storing the previous position of foot 0 and the previous orientation angle. Knowing the previous position of foot 0 and the orientation angle we can pick the closest alignment to the previous position.

5.2. Resolving ambiguity

The rigid dimensions of the hand held resolve most of the ambiguity problems. The remaining problems we handle by continuity of motion. As the user drags the hand held around the surface as well as rotates it, the position and orientation at time *t* is very similar to time *t-1*. In general we find orientation a stronger predictor than position, but we use both for more robust performance.

For a single user scenario there are only a few orientations that make sense. It is unlikely that a user will frequently want to turn the work upside down. In the single user case continuity of orientation works quite well. In a multi-user scenario more drastic rotations occur when sharing the work with someone seated at a different position. Continuity of motion still works in this scenario, but there are cases where it breaks. In a multi-user scenario, the Diamond Touch can sense each user individually. A default orientation range could be associated with each user. The active user would grasp the frame and ambiguity would be resolved using the identity of that user.

In our usage, continuity of motion has worked well. There are two failure cases. The first is if the user moves very fast so that the orientation between time *t* and time *t-1* is very different. This problem is readily resolved by processing of the points so that the sampling rate exceeds human hand speed. A more fundamental problem is when the user picks up the hand held (eliminating sensory contact) and then puts it down in a very different location and orientation. This is resolved through user instruction.

> "If you pick it up rather than slide it, it may get confused."

> "If it gets confused, pick it up, put it down in alignment with the image and then slide from there."

In our experience users had no difficulty in understanding the correct way to use the device and the work flow was not impeded.

5.3. Cases 2, 3 and 4

Cases 2, 3 and 4 all have a similar treatment. A discussion of case 4 will illustrate what is necessary. Given the 3 peaks in X and the two peaks in Y, in case 4, there are six possible locations for a conductive foot, as shown in figure 11. There are 120 possible assignments of the possible points to the three conductive feet. However, only a few of these possibilities will explain all of the data. Each foot must be assigned to a different X peak or one of the X peaks will be unexplained. Similarly both Y peaks must be used. For example, the combination a, b and d would leave the third X peak unexplained and therefore is not a valid combination.

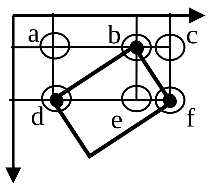

Fig 11. Points derived from X, Y peaks

For case 4 there are 6 possible point/foot assignments that use all of the histogram peaks. Given one of these possible assignments, we can test to see if the assignment is consistent with the known width and height of the handheld device. The distance between foot 0 and foot 1 must be equal to the handheld width. The distance between foot 1 and foot 2 must be equal to the height. This check will eliminate all but one possible configuration in cases 3 and 4. In some situations case 2 can have multiple configurations that are consistent with the data. The final ambiguity of case 2 is resolved using continuity of motion.

6. Display alignment

The handheld display and the context display on the touch table need to correspond in size and location. In XICE all coordinates are specified in *view independent coordinates* (VIC). When a display device is configured its pixels per inch are specified as well as the preferred viewing distance. This configuration information is sufficient to convert VICs into pixels. If the handheld and the touch table projector have their pixels per inch set correctly and have the same viewing distance then their coordinate systems are automatically consistent when application information is drawn in VICs. There are other systems besides XICE that account for variations in pixels per inch, but the viewing distance is also critical in this focus plus context technique. The resolution of the Diamond Touch for sensing the conductive feet can lead to some misalignment between the hand held and the table top. However, the hand held bezel provides enough of a break in visual continuity that this is not noticeable.

7. User experience

We have not done formal usability studies on Spilling. We have, however, learned some things by watching people try to use the system.

Our first observation was that fixed location widgets are problematic when interacting with a Spilling table. Using the non-dominant hand users freely scroll around their workspace. However, items like menus and tool bars that are at fixed locations require that the user scroll away from their work context to bring such widgets into the focus region of the hand held. The user must then scroll back to the original work area. This is very awkward. Similar problems have been noted with very large displays where substantial effort is required to move from a work location to some distant widget group. We have dealt with this problem by providing access to all widget actions through pop-ups at the work site.

A second issue is interactions that require dragging across a distance larger than the hand held's display size. At first we believed that this was something that just could not be done. That did not concern us because such techniques are very difficult with hand helds anyway. However, users quickly learned to hold the dragged item in the hand held view and then simultaneously scroll the work with the non-dominant hand. Once a user has seen the technique it works very well and is much better than trying to do the same thing on the hand held alone. The fact that the hand held, with larger mass and sliding friction, holds still while the work is moved is also more effective than dragging by moving the hand held as in Ubiquitous Graphics or Peepholes.

8. User interface security threats

We do not presume to deal with all possible security threats to network-connected computers. However, we have tried to avoid introducing any new threats when distributing the user interface to foreign machines. In Spilling there are three security threats in the user interface. The first and most dangerous is the acceptance of input from untrusted machines. Spilling resolves this by only accepting scrolling information and by isolating that input from the application, as shown in figure 1. The second is that a table top display server has access to all of the information being presented on the table. This threat is fundamental to distributed UI architectures and cannot be removed. Any information displayed by a computer for humans to see can be stolen by the display computer.

There is one more subtle UI attack that is possible. A rogue table could lie about scrolling information and about the context while scrolling a different part of the application under the hand held and thus deceiving the user about where their input is going. The table top cannot introduce any new interaction to the hand held, it can only scroll existing elements around. For such a threat to succeed there would need to be a deep understanding of the application by the table top and an application so designed that the user could be deceived by the context. This is a very obscure threat that is easily thwarted by appropriate application design.

9. Summary

We have created an architecture where users can bring hand held devices with small screens to a table top display and “spill” their user interface onto the table. The table provides a large context to the normal interactions of the hand held device. We have also shown how the hand held and the table can be visually synchronized. Lastly we have presented an algorithm for sensing hand held position and orientation through a projective sensing devices such as the Diamond Touch.

10. References

[1] Baudisch, P., Good., N., Stewart, P., “Focus Plus Con-text Screens: Combining Display Technology with Visualization Techniques”, *User Interface Software and Technology (UIST ’01)*, ACM (2001), pp. 31-40.

[2] Bederson, B. B., Grosjean, J., Meyer, J., “Toolkit De-sign for Interactive Structured Graphics.” *Software En-gineering*, IEEE (2004), pp. 535-546.

[3] Dietz, P.H.; Leigh, D.L., "DiamondTouch: A Multi-User Touch Technology." *ACM Symposium on User Interface Software and Technology (UIST ‘01)*, ACM (2001), pp 219-226.

[4] Lee, J. C., Hudson, S. E., Summet, J. S., and Dietz, P. H., “Moveable Interactive Projected Displays using Projector-based Tracking,” *User Interface Software and Technology (UIST ’05)*, ACM (2005), pp 63-72.

[5] Myers, B. A., “Using Handhelds and PCs Together”, *CACM*, 44(11), ACM (Nov 2001), pp 34-41.

[6] Olsen, D. R., Nielsen, S. T., and Parslow, D., “Join and Capture: a Model for Nomadic Interaction,” *User Interface Software and Technology (UIST ’01)*, ACM (2001), pp 131-140.

[7] Pering, T., Ballagas, R., and Want, R. “Spontaneous Marriages of Mobile Devices and Interactive Spaces,” *CACM*, 40 (9), (Sept 2005), pp 53-59.

[8] Richardson, T., Stafford-Fraser, Q., Wood, K. R., Hopper, A., "Virtual Network Computing", *IEEE Internet Computing*, 2(1), 1998.

[9] Sanneblad, J. and Holmquist L. “Ubiquitous graphics: combining hand held and wall-size displays to interact with large images.” *In Proceedings of AVI 2006*, ACM, (2006), pp 373-377.

[10] Scheifler, R. W. and Gettys, J. “The X Window System” *ACM Transactions on Graphics*, 5(2), (April 1986), pp 79-109.

[11] Shen, C., Vernier, F. D., Forlines, C., and Ringel, M. “DiamondSpin: an Extensible Toolkit for Around-the-table Interaction,” *Human Factors in Computing Systems (CHI ’04)*, ACM, (2004), pp 167-174.

[12] http://wacom.com/

[13] Want, R., Perins, T., Danneels, G., Kumar, M., Sundar, M., and J. Light. “The Personal Server: Changing the way we think about Ubiquitous Computing.” *Ubiquitous Computing* (UbiComp ‘02), Springer Verlag, (2002).

[14] Wilson, A. D., “PlayAnywhere: a Compact Interactive Tabletop Projection-vision System,” *User Interface Software and Technology (UIST ’05)*¸ ACM (2005), pp 83-92.

[15] Yee, K.-P. “Peephole Displays: Pen Interaction on Spatially Aware Handheld Computers.” *Human Factors in Computing Systems (CHI ’03)*, ACM, (2003), pp 1-8.