Mobile Spatial Tools for Fluid Interaction

Tobias Isenberg,[1] Simon Nix,[1] Martin Schwarz,[1] André Miede,[1] Stacey D. Scott,[2] Sheelagh Carpendale[1]
[1]University of Calgary, Canada
{isenberg | nixs | maschwar | amiede | sheelagh}@cpsc.ucalgary.ca
[2]University of Waterloo, Canada
sdscott@acm.org

Abstract

Fluid interaction techniques are increasingly important for effective work on interactive displays such as tabletops. We introduce mobile spatial tools to support such fluid interaction by affecting the properties of objects in the interface spatially rather than temporally. Our tools allow us to control multiple objects simultaneously using high-level, task-driven actions without the need for setting low-level properties of objects. We demonstrate a number of specific tools and their application in fluid interaction scenarios.

1. Introduction—Fluid Interaction

Tabletop displays with multi-touch capabilities have recently become a focus area in interface research. The effectiveness of such multi-person working environments can be increased by providing support for thinking on the task- and activity-level, shielding users from having to deal with low-level attributes of the interface objects such as size or orientation. This support can be realized by providing tools that avoid interruptions to the thinking processes and that allow people to focus on their task. We discuss how to support such *fluid interaction* by affecting object properties spatially rather than temporally. We apply these modifications through mobile spatial tools that support an intuitive exploration of, and interaction with the data.

Guimbretière et al. have convincingly argued this case for large, high-resolution, interactive wall displays in collaborative activities [4]. In particular, they stressed the need for simple yet fluid, preferably modeless, interaction techniques that are not interrupted by window elements, pop-up dialogs, and similar interaction metaphors known from desktop interfaces. The abrupt nature of such interaction elements can hinder effective work in collaborative settings because they are surprising to users who did not initiate them. Guimbretière et al. thus characterized fluid interaction as avoiding "continual interruptions to the flow of activity" in graphical user interfaces [4]. We agree with this definition and restate it positively saying that interface components assist fluid interaction if they support the users' high-level cognitive processes such as information organization without forcing them to explicitly specify low-level states of the objects. In addition, other authors stressed the importance of easy transitions between activities [10]. Providing fluid interaction, hence, means that people can pursue their goals and are supported by non-disruptive, understandable tools.

A number of techniques leverage familiar types of interaction to make it easier for people to achieve their goals. ZoomScape on wall displays [4] resizes objects depending on their vertical location rather than requiring the user to resize them explicitly. The Personal Digital Historian [11] automatically reorients and resizes objects w. r. t. their position on a circular tabletop layout. Experiences from this system led to the development of the DiamondSpin toolkit [12] that uses polar coordinates for the automatic rotation of information objects to address orientation issues on tabletop displays. Rotate'N Translate (RNT) [7] alters position and orientation through a simple passing motion rather than changing them one at a time. CrossY [2] supports pen-based interaction by using strokes that cross through the options that are to be selected. While facilitating easy transitions between consecutive commands, CrossY supports both fluid interaction by allowing users to interact on a high-level as well as the control of low-level object attributes. Gesture-based approaches, in general, such as multi-finger or whole hand actions [5, 13] as well as tool-specific gestures [8, 9] can increase the fluidity of interaction by directly incorporating elements from real-world interaction. This approach can be extended even further by using physical simulation and modeling the interface after the real world [1].

2. Mobile Spatial Tools

Most of these fluid interaction techniques rely on moving objects to a different location on the interface in order to initiate the desired modifications, thus the effects are based on the spatial location of the object on the interface. With the addition of mobility we extend the spatial character of previous fluid interfaces. Instead of moving interface objects to the spatial location where the desired interface action occurs, we move the spatial effects of the desired interface action to the objects we want to affect. We thus extend fixed spatial interface actions that are advantageous for fluid interaction with independent, spatially explicit tools that cause local effects. Since our tools act through their movement over the interface, influencing its properties based on their spatial location, we think of them as *mobile spatial tools*.

Our goal in creating these tools is to support high-level social and task activities, while minimizing the computer-related actions required. For example, the tools allow users to concurrently affect and control numerous objects (such as images and text documents; other types are also possible), without the need to select any of them specifically or requiring users to invoke a menu selection. To explain how this concept is manifested in our example tools we introduce them based on the high-level activities they support.

Motion tool: In situations when the display becomes covered with too many objects it is sometimes desirable to clear space for the next stages in the work process. Instead of having to perform low-level activities such as select and move, or even group-select and move, it may be much easier to simply sweep the area clean. For this type of interaction we provide a tool that moves objects out of its influence range as shown in Fig. 1. The same tool, used stationary, can also be employed for the task of keeping an area clean of objects. This may be used to keep interface elements accessible as shown in Fig. 1(b).

Piling tool: Also when dealing with large numbers of interface objects, it may become necessary to group a number of selected elements. While this may not be necessary as a preparation to move or remove objects, it may be helpful to be able to gather them into a neat package or stack that takes little space. We support this task with a tool that pulls the objects in its range into its center as shown in Fig. 2.

Magnification tool: While informally browsing through a collection of objects it is often desired to get a closer look at some of the objects. We support this action with a local magnification tool. Since it is commonly known from the detail-in-context literature, the mobile magnification interaction effect will be familiar to most readers. This tool allows running one's finger (or pen/mouse cursor) over the objects, magnifying those that the finger touches (Fig. 3).

Alignment tool: The inherent orientation of many document types may make it necessary to align such objects. We support such interactions with a tool that temporarily reorients objects within a region, as shown in Fig. 4.

Motion orientation tool: In other situations it may be-

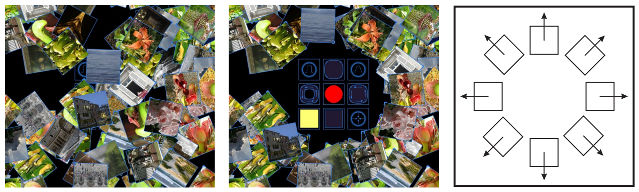
(a) Without tool. (b) With motion tool. (c) Schematic.

Figure 1. Using a motion tool to clear space or to keep active interface areas accessible.

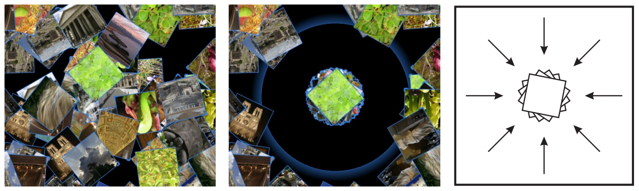
(a) Without tool. (b) After application. (c) Schematic.

Figure 2. Tool to collect objects into piles.

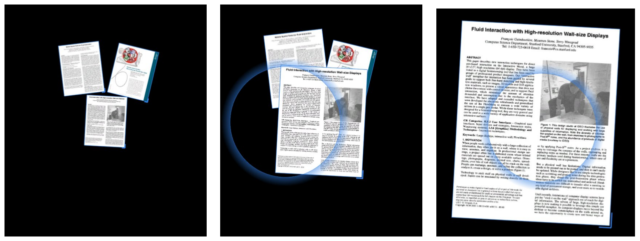
(a) No resizing. (b) Starting to resize. (c) Fully magnified.

Figure 3. Resizing for closer examination.

come necessary to do a similar object alignment permanently. This can be achieved with a tool that derives the orientation from its motion by aligning the objects parallel or perpendicular to its path (Fig. 5). This tool supports, for example, casual combing through objects to line them up.

Radial orientation tool: In other situations radial orientations may be useful. Radial patterns can orient objects either inward (Fig. 6) or outward (Fig. 7). Consider the the example in Fig. 7 where radial orientation was used in a personal work territory to align documents for individual work.

3. Realizing Mobile Spatial Tools

Our mobile spatial tools allow users to affect object properties spatially and based on fluid, high-level tool movement. We realize this by mapping a tool's position to the low-level properties of the affected objects. Our approach is based on a spatial representation of the interface's properties [6], sampled on a regular grid as interaction buffers. Interface objects look up these properties by querying the interaction buffers (Fig. 8(a)). Using tools to make local changes to the sampled interface properties, we can adjust all objects located over such a modified region to use the new values and change their behavior accordingly (Fig. 8(b)).

The spatial tool approach relates to Magic Lenses [3] which allow users to interact with different layers of the (see-through) interface and to locally show different representations of objects within the lens region. Magic lenses act as filters that locally allow interaction with other aspects of the interface or locally modify the way objects are displayed. While mobile spatial tools can be used similarly to magic lenses in that they can affect representation properties, they can also be used to affect object actions such

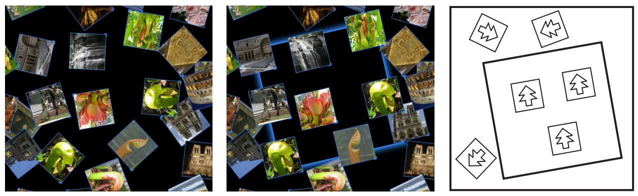
(a) Without tool. (b) After application. (c) Schematic.

Figure 4. Tool that temporarily rotates objects to line them up with the tool's orientation.

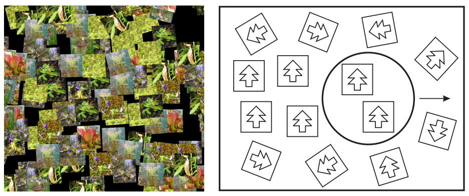
(a) Example. (b) Schematic.

Figure 5. Tool that re-orients permanently and perpendicular to the tool's motion vector.

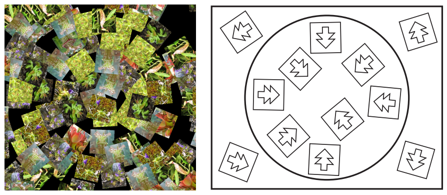
(a) Example. (b) Schematic.

Figure 6. Tool for radial re-orientation w. r. t. the tool for collaborative work on tabletops.

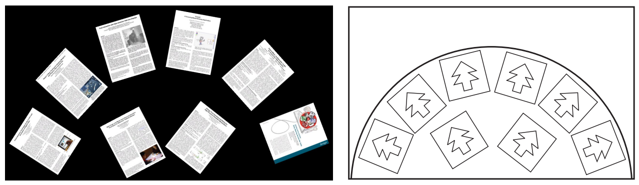
(a) Example. (b) Schematic.

Figure 7. Aligning objects for individual work.

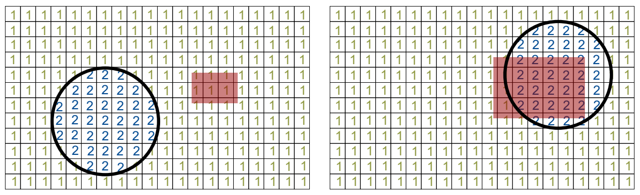
(a) Before object is affected. (b) After object is affected.

Figure 8. Affecting the size of an interface object (red rectangle) with a mobile spatial tool (circle) via interaction buffers.

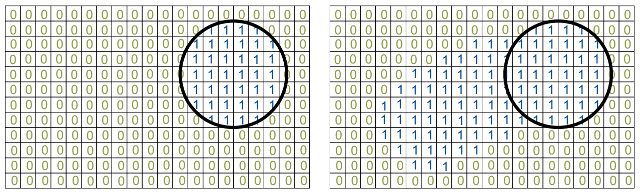
(a) Instantaneous buffer: changes are only applied within the current tool coverage. (b) Persistent buffer: changes remain in the interaction buffer after the tool is moved.

Figure 9. Two types of interaction buffers.

as various types of motion and orientation. In addition, they are single-action tools neither requiring two hands nor using a see-through-and-click approach.

3.1. Tool Characteristics

In addition to the specific property that is being affected, the tools' characteristics are governed by a number of aspects including temporality, mobility, shape, attenuation, order, and creation which we discuss individually below.

Temporality: Tools can make persistent changes or can have temporary effects, depending on the way the buffers are refreshed. To realize this difference we support two different types of buffers: *instantaneous* and *persistent* buffers (Fig. 9). Instantaneous buffers (Fig. 9(a)) are always initialized with a default value and changes are only applied and maintained for one simulation/rendering step. These buffers allow us to model temporary changes that are only in effect while a tool is being used at a specific location (e. g., a relative size tool that operates like a magnification lens). In contrast, persistent buffers (Fig. 9(b)) keep all changes once they have been applied. These buffers allow us to model permanent changes to the properties of the interface objects at that location. This difference in buffer interaction also gives rise to two different ways of using tools in an interface. Using instantaneous buffers, tools are continuously existing entities/widgets that are created, moved, and deleted with explicit actions. This differs from traditional tools that are controlled through toolbars because several mobile spatial tools can exists at the same time and affect objects simultaneously. Also, their actions are not permanent as objects return to their previous states when the tools are removed. Alternatively, with persistent buffers which continuously affect objects, tools only exist while the interface is touched and, thus, do not require moving tools across the surface.

Mobility: Our tools are generally mobile and are moved over the interface to initiate actions. However, tools can also be left at a static position while objects are affected when moving them over the tool's influence range. The latter type of use can model, among others, fluid interaction metaphors such as ZoomScape [4] as well as aspects of the Personal Digital Historian [11] and the DiamondSpin toolkit [12].

Shape: Our tools are flexible in their shape as it is only necessary for them to have a local spatial representation to modify a small portion of the interface's global state. While it is easiest to use mathematically simple shapes such as circles and rectangles, tools are by no means limited to these. They can be configured in size and shape to fit the user's needs, ranging from a few pixels to covering the screen.

Attenuation: Depending on their intended use, our tools can have both a stepped or an attenuated effect. While it may make sense to use attenuation when controlling properties such as size, non-attenuated tools may be useful for affecting properties such as orientation.

Order: If our tools are used mainly to examine objects which do not require additional interaction, then the tools are kept as the top-most objects (see, e. g., the magnifying tool in Fig. 3). This ensures the tool can always be reached for further touch-and-drag interaction or for deleting it. In cases where the tools have to allow interactions with the objects that are affected, we keep the tools always behind other objects (e. g., Fig. 1 and 4). This ensures that affected objects can be touched and interacted with at all times.

Creation: Tools are usually created explicitly using a number of buttons (Fig. 1(b)). They can then be moved by touching them and dragging them over the interface. If not needed anymore, they are deleted by dragging them onto a special erasing area (in the center of Fig. 1(b)).

3.2. Implementation Aspects

Based on a buffer stack architecture [6], we assign each surface with interface objects a stack of interaction buffers to control object properties. These surfaces include the interface background as well as independent container widgets. Mobile spatial tools are treated as regular objects and are connected to a surface. Therefore, they can access the associated buffers to apply changes. All objects subjected to buffer control similarly access the buffers, read from them, and act accordingly. The system is implemented in OpenGL while buffer access and management are independent from the rendering. It is able to control 2,000 objects and more at interactive rates on modern PC and graphics hardware.

As buffer writing for large areas can become computationally expensive, tools only update their associated buffers if necessary. For instantaneous buffers we implemented an updating scheme that only refreshes the buffer if notified by a tool. In addition, we locally cache tool buffer values so that tools do not need to recompute their usually relatively complex buffer stencil for each update.

4. Conclusion

We introduced interaction by means of mobile spatial tools. Similar to previous techniques, our tools control the behavior of objects based on spatial location. The mobility of our tools, however, makes it possible to flexibly and fluidly control objects using intuitive tool placement and movement. The local manipulation of spatially maintained attributes allows us to design and interactively change the interface's behavior by placing functionality where needed, the location of which may change over time. This removes the need for having to deal with object properties directly, enables users to pursue their interaction goals without interruptions, and allows them to easily transition between activities.

Acknowledgments

We thank our funding agencies Alberta Ingenuity, CFI, iCORE, NSERC, and SMART Technologies as well as Petra Neumann, Mark Hancock, and Jens Grubert.

References

[1] A. Agarawala and R. Balakrishnan. Keepin' It Real: Pushing the Desktop Metaphor with Physics, Piles and the Pen. In *Proc. CHI*, pp. 1283–1292, New York, 2006. ACM Press.

[2] G. Apitz and F. Guimbretière. CrossY: A Crossing-Based Drawing Application. In *Proc. UIST*, pp. 3–12, New York, 2004. ACM Press.

[3] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose. Toolglass and Magic Lenses: The See-Through Interface. In *Proc. SIGGRAPH*, pp. 73–80, New York, 1993. ACM Press.

[4] F. Guimbretière, M. Stone, and T. Winograd. Fluid Interaction with High-Resolution Wall-Size Displays. In *Proc. UIST*, pp. 21–30, New York, 2001. ACM Press.

[5] M. Hancock, S. Carpendale, and A. Cockburn. Shallow-Depth 3D Interaction: Design and Evaluation of One-, Two- and Three-Touch Techniques. In *Proc. CHI*, pp. 1147–1156, New York, 2007. ACM Press.

[6] T. Isenberg, A. Miede, and S. Carpendale. A Buffer Framework for Supporting Responsive Interaction in Information Visualization Interfaces. In *Proc. C^5*, pp. 262–269, Los Alamitos, CA, 2006. IEEE Computer Society.

[7] R. Kruger, S. Carpendale, S. D. Scott, and A. Tang. Fluid Integration of Rotation and Translation. In *Proc. CHI*, pp. 601–610, New York, 2005. ACM Press.

[8] G. Ramos and R. Balakrishnan. Fluid Interaction Techniques for the Control and Annotation of Digital Video. In *Proc. UIST*, pp. 105–114, New York, 2003. ACM Press.

[9] G. Ramos and R. Balakrishnan. Zliding: Fluid Zooming and Sliding for High Precision Parameter Manipulation. In *Proc. UIST*, pp. 143–152, New York, 2005. ACM Press.

[10] S. D. Scott, K. D. Grant, and R. L. Mandryk. System Guidelines for Co-located, Collaborative Work on a Tabletop Display. In *Proc. ECSCW*, pp. 159–178, Norwell, 2003. Kluwer.

[11] C. Shen, N. Lesh, and F. Vernier. Personal Digital Historian: Story Sharing Around the Table. *interactions*, 10(2):15–22, Mar./Apr. 2003.

[12] C. Shen, F. D. Vernier, C. Forlines, and M. Ringel. DiamondSpin: An Extensible Toolkit for Around-the-Table Interaction. In *Proc. CHI*, pp.167–174, New York, 2004. ACM Press.

[13] M. Wu and R. Balakrishnan. Multi-Finger and Whole Hand Gestural Interaction Techniques for Multi-User Tabletop Displays. In *Proc. UIST*, pp. 193–202, New York, 2003. ACM Press.