Mobile Spatial Tools for Fluid Interaction

Tobias Isenberg,[1,2] Simon Nix,[2] Martin Schwarz,[2] André Miede,[2] Stacey D. Scott,[3] Sheelagh Carpendale[2]
[1]University of Groningen
isenberg@cs.rug.nl
[2]University of Calgary
{nixs|maschwar|amiede|sheelagh}@cpsc.ucalgary.ca
[3]University of Waterloo
s9scott@uwaterloo.ca

1. Fluid Interaction

Large displays with multi-touch capabilities have recently become a focus of interface research. The effectiveness of such *multi-person* working environments can be increased by providing support for thinking on the task and activity level, freeing users from having to deal with low-level attributes of the interface objects such as size or orientation. This support can be better realized by providing tools that avoid interruptions to the thinking processes, thus allowing people to focus on their task. We present use of i-buffers [2] to unify creation of such *fluid interactions* by specifying interface object properties spatially rather than temporally. Modifications can be applied through mobile spatial tools—visible local and movable tools that are available in the shared, collaborative workspace [1]—to support an intuitive exploration of and interaction with data.

Previously, interaction techniques termed fluid usually relied upon either integrated interaction or screen location specific interaction to initiate desired modifications. We extend the spatial character of previous fluid interfaces through screen location specific interaction with surfaces, which can either be stationary or mobile. Instead of requiring movement of interface objects to the a stationary location where the desired interface action occurs, we explore moving the interaction effects to the objects we want to affect. We thus extend spatially fixed interface actions that are advantageous for fluid interaction with independent, spatially explicit tools that cause local effects. As our tools act through their movement over the workspace or their current workspace position, we call them *mobile spatial tools*.

2. Mobile Spatial Tools

Our framework uses i-buffers, a concept similar to z-buffers that are based on a buffer stack architecture [2], which was initially designed to provide better interactive rates for large high-resolution displays. Using this buffer stack architecture, we assign each surface with interface objects a stack of i-buffers to control object properties. These surfaces include the interface background as well as independent container widgets. *Mobile spatial tools* are objects that are connected to a surface and can access the associated surface buffers to affect interactions and to apply changes.

To realize mobile spatial tools in software, we sample a spatial representation of the interface's properties on a regular grid as i-buffers. Interface objects look up these properties by querying the i-buffers. Using tools to make local changes to the sampled interface properties, all objects located within a region can read the new values and change their behavior accordingly. To simplify i-buffer writing, associated i-buffers are only updated if necessary.

An i-buffer implemented spatial interface tool is governed by a number of aspects which are discussed below.

Temporality: Tools can make persistent changes or can have instantaneous effects, and thus affect way the i-buffers are refreshed. Tools with instantaneous effects 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 exist 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 changes which *continuously affect* objects, tools only exist while the interface is touched and, thus, do not require moving tools across the surface.

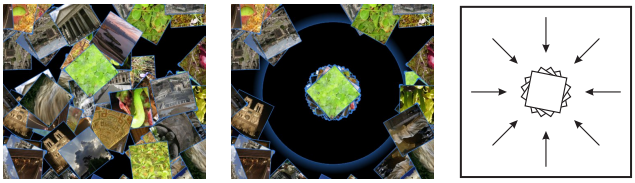
Mobility: Our tools are generally mobile and are moved over the interface to initiate actions. However, tools can also be left at a static position while objects are affected when moving them over the tool's influence range.

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 either 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.

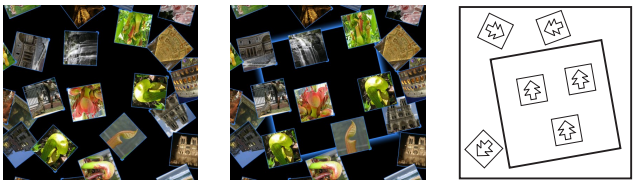
Order: If our tools are used mainly to examine objects which do not require additional interaction, then the tools are kept as the top-most objects (see, e. g., the magnifying tool in Fig. 1(c)). This ensures the tool can always be reached for further touch-and-drag interaction. In

(a) Clearing space or keeping active interface areas accessible.

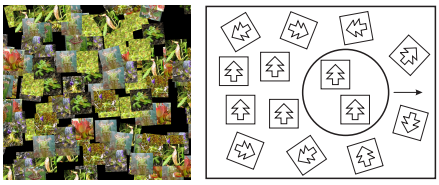


(b) Collecting objects into piles.

(c) Resizing for closer examination.

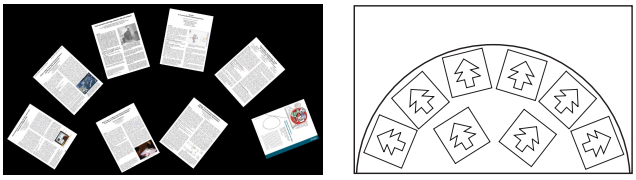


(d) Temporarily rotating objects to line them up.



(e) Re-orienting permanently & perpendicular to the motion vector.

(f) Radial re-orientation for collaborative work on tabletops.



(g) Aligning objects for individual work.

Figure 1. Mobile spatial tools.

cases where the tools allow interactions with the objects they affect, the tools are kept behind the other objects (e. g., Fig. 1(a) and 1(d)). Thus, affected objects can be touched and interacted with at all times.

We designed the tools to support high-level social and task activities, while minimizing the interface related actions required. The tools allow users to concurrently affect and control numerous objects (images, text) *without* having to select any of them specifically or requiring a menu selection. In contrast to interactions that first require a selection and then apply a function, our tools provide localized interaction, helping to improve workspace awareness by providing consequential communication in direct-touch environments. Below we introduce our tools based on the high-level activities they support.

Motion tool: In situations when the display becomes cluttered 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 group-select and move, it may be easier to simply sweep the area clean. A tool can simply push objects out of its influence range. The same tool, used stationary, can be employed for the task of keeping a given area such as interface elements (buttons, menus) clean of objects and accessible (Fig. 1(a)).

Piling tool: When dealing with large amounts of workspace content, it may be necessary to group a number of elements by gathering them into a neat package or stack that takes little space. The piling tool pulls the objects in its range into its center (Fig. 1(b)).

Magnification tool: While informally browsing through a collection of objects it is often desired to get a closer look at some of them. Our local magnification tool allows running one's finger (or pen/mouse cursor) over the objects, magnifying those that the finger touches (Fig. 1(c)).

Alignment tool: Many virtual objects are orientation-dependent (text, graphics), re-aligning such objects in a shared workspace facilitates comprehension. We support this with a tool that temporarily reorients objects (Fig. 1(d)).

Motion orientation tool: In other situations it may be 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. 1(e)).

Radial orientation tool: In other situations radial orientations may be useful. Radial patterns can orient objects either inward for collaboration (Fig. 1(f)) or outward (Fig. 1(g)) for individual work.

References

[1] B. B. Bederson, J. D. Hollan, A. Druin, J. Stewart, D. Rogers, and D. Proft. Local Tools: An Alternative to Tool Palettes. In *Proc. UIST*, pp. 169–170, New York, 1996. ACM Press.

[2] 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.