A Malleable Physical Interface for Copying, Pasting, and Organizing Digital Clips

Florian Block, Nicolas Villar and Hans Gellersen
Computing Department, Lancaster University
InfoLab21, Lancaster University, LA1 4WA Lancaster, UK
{block, villar, hwg}@comp.lancs.ac.uk

ABSTRACT
We present a system that extends a typical workstation environment with a malleable physical interface for working with digital clips. It allows users to pick digital clips, give each its own dedicated key for direct access, and combine keys dynamically on a physical surface in a way that inherently reflects the state of an extended clipboard. The system affords copying and pasting of multiple clips each directly accessible through its own key shortcut. The keys can also be dynamically re-arranged to organize clips, and taken from workstation to another to transport clips, acting simultaneously as token and as copy-paste-interface for a digital object.

Author Keywords
Physical Interfaces, Clipboard Extension.

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
Since the introduction of Xerox Star in 1981, the concept of data exchange between multiple running applications has become commonplace in modern operating systems. In fact, it has become a standard mechanism between most applications and operating systems in the form of the system clipboard, universally accessible by dedicated shortcuts to copy and paste a clip (e.g. Ctrl+C, Ctrl+V). A great benefit of the standardized clipboard is the transparency that comes from having these dedicated keyboard shortcuts, which allow a single clip to be created and recalled by simple actions. However, system clipboards are usually limited to dealing with a single item of information at a time.

Standard system clipboards can be extended by one of any third-party software clipboard extensions, which have as primary goal to allow the user to manage multiple clip objects at the same time. Having to deal with multiple items inevitably requires the user to resort to more complex sequences of mouse or keyboard input in order to manage the various clips they create and retrieve, with the effect that the clipboard is brought to the foreground of attention. Other common clipboard extensions aim to allow clipboard data to be shared amongst multiple machines or networked users, likewise to the effect that usage becomes considerably less transparent.

In this paper we present a design that preserves the ease of use of a single-item clipboard while allowing the user to manage multiple clips of information. We do this by introducing a new interface device, alongside the mouse and keyboard, to which we factor out the controls for copying, organizing and retrieving clipboard. In addition, our system allows a mechanism for transporting clips of information between different networked machines.

Our design is based on the concept of *malleable physical interfaces* – devices made up of basic controls (such as buttons) that a user can arrange in accordance with their task. In our design we foresee that users can introduce a dedicated control for *each* clipboard item they generate. We maintain the simplicity of having a dedicated shortcut for a single clip (as found in the conventional system clipboard) but we are able to have this for *multiple* clips (cf. Figure 1).

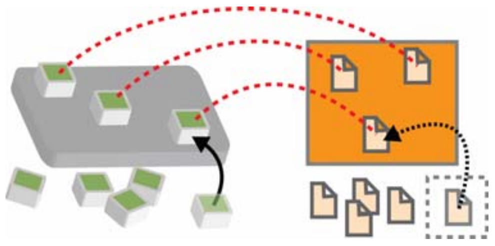

Figure 1: Each clip has its own dedicated key control.

Figure 2 illustrates our realization of this concept: a surface is deployed alongside the traditional input devices, on which additional keys can be arranged.

As in existing practice, a user selects an object for copying by selecting it in their GUI. Attaching a key to the surface at this time has the effect, in a single step, of generating a new clip and of introducing the key as its control. Keys can be rearranged and clustered into meaningful groups and removed to the effect of also removing the corresponding clipboard item from the computer. Clips can also be exchanged between two systems by moving keys between surfaces. In order to retrieve the content into an application, the user simply presses the appropriate key.

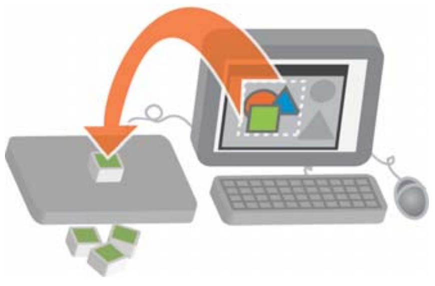

Figure 2: Extending the desktop environment with a physical interface for working with clips.

RELATED WORK

Clipboard extensions are widely available for standard windowing systems. In a research context, related work has considered synchronized copying and pasting across multiple computers [4], intelligent adaptation of clips to the context into which they become pasted [5], web-based working with clips [2], and tools optimized for particular practices such as programming [8]. Our work is more specifically focused on easing work with multiple clips but also facilitates work across multiple computers with a web-based clip repository, and with physical keys as unique handles to clips, addressing the problem of synchronization.

The concept of associating digital objects with physical tokens or icons (“phicons”) for tangible manipulation has also been explored widely. Ullmer’s mediaBlocks, for example, demonstrated the use of physical objects (wooden blocks) for copying digital objects from one device to paste it on another device [6]. We provide the same support for transporting digital data with a physical key but provide a complete system integrated with existing clipboard mechanisms. Conceptually we are more concerned with efficient interfacing of clips than their embodiment for tangible interaction.

A key aspect of our work is the provision of a customised physical interface that users can adapt to their task. Greenberg and Boyle presented a system that lets users map graphical controls to physical interface objects (“phidgets”) [3], a mechanisms that for example allows frequently used functions to be mapped to dedicated ‘function keys’. In this analogy, our interface provides a single dedicated function key for copy&paste, operations but we allow as many replicas as a users needs for working simultaneously with multiple clips. Villar et al. introduced the VoodooIO toolkit providing interactive surfaces on which users can freely add, manipulate and remove controls such as buttons, dials and sliders [7]. We adopt their work as platform for our interface, as it makes adding and removing of controls very seamless, in terms of effort (a control has a simple pushpin connector) as well as speed (a new control is detected instantaneously, allowing immediate system response).

INTERACTION DESIGN

Our system assumes a standard workstation environment with a windowing operating system, and extends this with a surface and set of keys that can be attached on the surface.

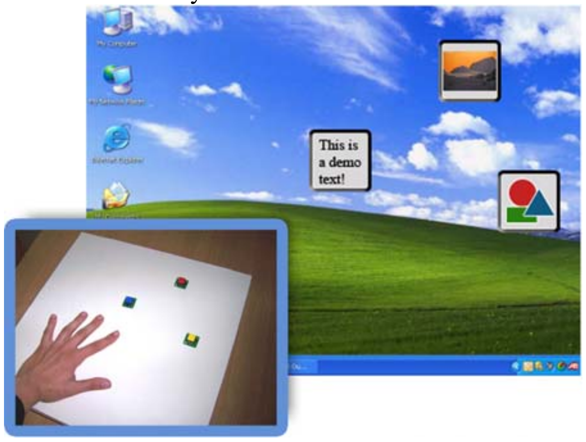

Figure 3: The clipboard content can be viewed by pressing on the surface, with thumbnails shown in a layout corresponding with the spatial layout of the physical interface.

The surface and the keys attached at any time can be thought of as a custom keyboard, attached as additional input device to the workstation, dedicated to working with clips. Based on this setup, our system supports the following clipboard operations, as seen from the user’s perspective:

- *Copying:* To copy an object, the user selects it in the GUI, and then attaches a key to the surface. A digital copy of the selected object is made and associated as clip to the key. For user feedback, a thumbnail of the newly generated clip appears on the screen at coordinates relative to the position of the key on the surface. After a few seconds the thumbnail fades out. The user can create multiple clips by repeating the process.
- *Organizing:* The user can organize clips by rearranging the layout of the keys on the surface. For instance, clips related to a specific topic can be grouped by corresponding arrangement of keys. To support the user in recalling key-and-clip associations, the user can press down on the surface, which causes all clip thumbnails to be shown on the screen, in a layout corresponding to the layout of keys on the surface (cf. Figure 3). The thumbnails are shown for as long as the user presses on the surface, and then fade out.
- *Pasting:* By pressing a key, a copy of the associated clip is pasted into the current application’s context. During this process the association is maintained which makes it possible to paste the same clips multiple times.
- *Transporting:* It is possible to transport a clip between different workstations by moving the corresponding key

from the surface of one computer to that of another. The clip is then removed from the source workstation and made locally available on the target.

- *Deleting:* The user can also dissociate clips by detaching the corresponding key. Unless the key becomes re-attached on the surface (or another computer's surface) within a specified period of time, it loses its association. A key that has become dissociated can be associated with a new clip at any time.
- *Maintaining Persistency:* The clips of all keys attached to a surface are persistent. This means, the information will be available to the user even after a system restart.

SYSTEM IMPLEMENTATION

Our implementation extends standard PCs with a VoodooIO surface and several VoodooIO buttons as keys. Like VoodooIO controls in general, the keys have a coaxial pushpin connector for attachment on the surface, and a universally unique ID (UID) for their identification. The pin connectors keep the keys firmly in place on the surface, and supply them with power and network connectivity through the surface to the PC. The surface is further augmented with sensors to detect pressure on the surface, and to register the position of a key when it becomes attached [1].

A *clipboard service* on the PC interfaces with the VoodooIO API, to receive VoodooIO system events (*New Key Added*, *Key Pressed* and *Key Removed*). The service further has system-level access to the standard system clipboard, and is able to receive and emulate Windows clipboard events. Each *clipboard service* registers with a central *clipboard server* via a standard HTTP protocol. Together, these components provide the following system functionality:

- *Associating Clip with Keys:* The *clipboard service* hooks into the standard system clipboard to monitor and access its functionality. When a key is attached to the surface, the service triggers a clipboard-copy event, to copy the current selection into the system clipboard. This data is intercepted, producing a separate copy which is bound to the UID of the key. A background task takes the copy and uploads it to the *clipboard server* along with the associated key UID. This enables the system to retrieve clipboard data, either locally (e.g. after a restart) or remotely when the key with the associated UID is pressed.
- *Reviewing Clip Contents:* The *clipboard service* creates a thumbnail for each clip, representing its content. If the system detects that the surface is pressed it displays the thumbnails of all keys that are currently attached to the surface, overlaying the workspace.
- *Retrieving clips:* When a key is pressed, its associated clip data is copied back into the system clipboard. The *clipboard service* then triggers a system-level clipboard paste event resulting in the appropriate clip being inserted into the current application context.
- *Moving / Transporting / Deleting Clips:* When a key is removed from a surface, the service stores a timestamp of the removal event and informs the *clipboard server* of the event and where it was registered. This allows the server to keep track of the last known location of each key. If the key is re-inserted on the same surface within a specified time period (20 sec. in our prototype) it retains its clip association, thus allowing keys to be rearranged without loss of information. However if the key is instead attached to a *different* surface – independently of the time between the two events – it is assumed that the intention is to transport the clip to a new workstation. In this case the *clipboard service* of the target computer uses the key's UID to retrieve the clip data from the *clipboard server*. In the remaining case, when a key is re-inserted on the same surface after a 'timeout', this is interpreted as re-using the key for a new association (to the object presently selected by the user at this point in time), and the previous association becomes overridden.

DISCUSSION

The presented design aimed to extend the functionality of the system clipboard without introducing additional complexity or overhead into the current practice. Our discussion of the efficacy of our design is based around the illustration shown in Figure 4. In the left column we illustrate the mechanism for using the standard system clipboard. Some of the controls of the standard input devices are given the dedicated role of copying and pasting clips to applications via the universal clipboard mechanism.

The middle column in Figure 4 shows an approach where the clipboard is extended with an additional software program (cf. "ECB", Extended Clipboard). For the ECB to implement support for multiple clips, it must by necessity introduce additional interaction mechanisms that may interfere with a user's control of an application. For example, the user must be able to organize and select the different clips. This can result in a further overloading of the input devices (e.g. by introducing additional keyboard shortcuts on the already limited set of keys) or, alternatively, in forcing the user to interact with an additional GUI, taking away the focus of the current application and implying a context-switch.

By introducing a dedicated device for dealing with clips we remove the need to overload existing input devices with additional functionality. Furthermore, since all of the management of clip data can be carried out by direct, spatially-multiplexed actions in the physical domain, we unburden the user from having to switch their application context to carry out these tasks. By using a *malleable* interface we are able to maintain the one-to-one relationship between keys and clips which makes the standard clipboard so easy to use. Our interface is able to extend and shrink to provide *exactly* the amount of direct input which is necessary for dealing with any particular number of clips at any given point in time.

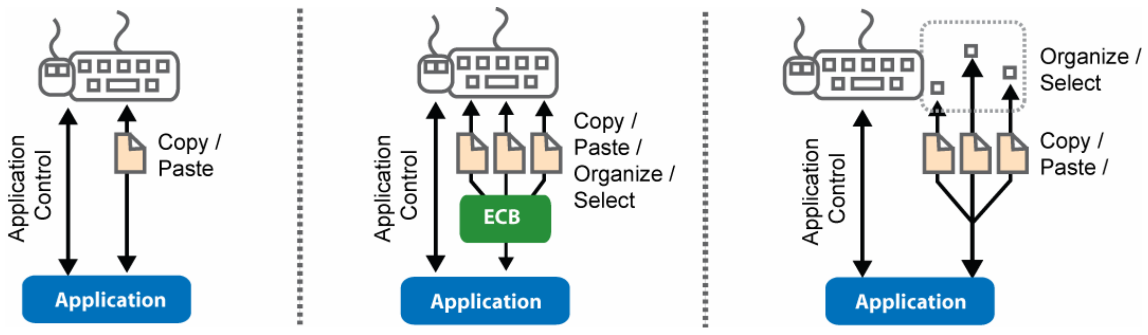

Figure 5: Conceptual views of the standard system clipboard (left), software-extended clipboard (middle) and a clipboard extension via a malleable interface (right).

Not only can keys be added or removed, but also arranged in a meaningful way. This ability can be used, for example, as a memory aid in grouping keys according to the meaning of the underlying clips; improving clip accessibility (e.g. commonly accessed items can be made more accessible); or inclusively as a way to improve the ergonomic layout of the interface. At any time the user can obtain an overview of the arranged clips with a simple gesture and a visual overlay that does not change the focus of the current application or overload the keyboard or mouse.

Another benefit of the malleability particular to the VoodooIO technology is the ability to move keys between surfaces. The ability to decouple keys from surfaces provides a natural way for transporting information between machines. The clip information is, effectively, embodied by the key which provides its interface. Having a physical item to represent a digital clip of information addresses the problem of synchronicity described in [5] when working with shared clipboards. Since VoodooIO components are uniquely identifiable we envision that in addition as acting as *tokens* and *interfaces* to clips, in future implementations keys could also act as *security keys* – making data only accessible on authorised workstations.

Due to the effortless way in which VoodooIO keys can be brought in and out of surfaces, or moved between surfaces, we are able to explore interaction scenarios where the concept of interface presence (both temporal and spatial) acts as an additional input modality. Even though each key contains only a binary input element - a button which can be pressed, mapped to the action of pasting a clip - we are able to encapsulate all the other actions necessary to work with clips (copying, organizing, transporting) to the different ways in which the key is made present to the system: on a surface, off a surface, somewhere on a surface, on one surface after another.

REFERENCES

1. Block, F., Gellersen, H., Hazas, M., Molyneaux, D. and Villar, N. Locating Physical Interface Objects on Interactive Surfaces. *Proc. Workshop on Mobile and Embedded Interactive Systems (MEIS'06)*, Dresden, Oct. 2006, Lecture Notes in Informatics, Springer –Verlag.
2. Dix, A., Catarci, T., Habegger, B., loannidis, Y., Kamaruddin, A., Katifori, A., Lepouras, G., Poggi, A., and Ramduny-Ellis, D. 2006. Intelligent context-sensitive interactions on desktop and the web. *Proc. Intl. Workshop on Context in Advanced interfaces*, AVI '06, Venice, Italy, May 2006, ACM Press, New York, 23-27
3. Greenberg, S. and Boyle, M. Customizable physical interfaces for interacting with conventional applications. *Proc. 15th Annual ACM Symp. on User interface Software and Technology* (UIST '02), Paris, France, Oct. 2002 ACM Press, New York, 31-40.
4. Miller, R. C. and Myers, B. A. Synchronizing clipboards of multiple computers. In *Proc. 12th Annual ACM Symposium on User interface Software and Technology* (UIST '99), Asheville, NC, USA, Nov. 1999, ACM Press, New York, 65-66.
5. Stylos, J., Myers, B. A., and Faulring, A. Citrine: providing intelligent copy-and-paste. In *Proc. 17th Annual ACM Symposium on User interface Software and Technology* (UIST '04), Santa Fe, NM, USA, Oct. 2004, ACM Press, New York, 185-188
6. Ullmer, B., Ishii, H., and Glas, D. mediaBlocks: physical containers, transports, and controls for online media. In *Proc. 25th Annual Conference on Computer Graphics and interactive Techniques* SIGGRAPH '98. ACM Press, New York, 379-386.
7. Villar, N., Gilleade, K., Ramduny-Ellis, D. and Gellersen, H. VoodooIO Gaming Kit: A real-time adaptable gaming controller. *Proc. of ACM Intl. Conf. on Advances in Computer Entertainment Technology (ACE 2006)*, Hollywood, June 2006, ACM Press.
8. Wallace, G., Biddle, R., and Tempero, E. 2001. Smarter cut-and-paste for programming text editors. Proc. 2nd Australasian Conf. on User Interfaces, Queensland, Australia, Jan. 2001. ACM Intl. Conf. Proceeding Series, vol. 14, 56-63.