notion of lenses or filters that can be manipulated on the data surface to change the appearance of objects seen within the lens. In addition, Lucas et. al. [6] uses a kind of local tool in their Visage system by allowing the user to place a dynamic query slider directly on the data surface. Finally, Meyer [5] demonstrated some examples of local tools that were generated on the fly by sketching in his Etcha-Pad system.

PAD++ LOCAL TOOLS

The current implementation of local tools in Pad++ is a preliminary investigation designed for the KidPad application previously mentioned (Figure 1). We have implemented local tools for drawing, erasing, selecting, creating hyperlinks, and for copying clip-art out of a scrapbook. In addition, there is a special toolbox for organizing tools.

The current user interactions with local tools are as follows. When KidPad starts up, the tools are lined up on the bottom of the screen. The toolbox is fixed at the bottom right. When the user pans or zooms, the tools stay on the surface, and thus pan or zoom with everything else. The toolbox, however, stays in the bottom right-hand corner of the screen. Clicking on the toolbox brings back the tools from wherever they were left on the Pad++ surface and lines them up on the bottom of the screen. Clicking on a tool picks it up so it can be used. Double-clicking drops the tool so another tool can be picked up.

The scrapbook has special interactions. It is a slider that shows the items in the scrapbook one at a time as the slider is moved. Clicking on an item in the scrapbook with the selection tool automatically makes a copy of the item that can then be positioned anywhere on the work surface.

USER TESTING

We tested the local tools in KidPad at Lowell Elementary School in Albuquerque, New Mexico with two fourth grade classes on an ongoing basis. We tested KidPad one hour a week for eight weeks (for each class). We rotated through the students, asking all 40 students to use KidPad in repeated tasks. We also tested KidPad at CHI'96 in the new CHIKids program, where 12 children aged 4-13 used KidPad for two hours.

Through this experience, we learned that local tools work. Even very young children learned to use them very quickly with minimal instruction, and many of them reported preferring local tools over the tool palette.

We also received valuable feedback from the kids about the tools usability. The biggest problem is that children (and many adults for that matter) have a difficult time double-clicking to drop the tool. We plan to make it possible to drop a tool with a single click on the toolbox. Also, a single click on any other tool will drop the first one, and pick up the second one.

The lack of local tools for panning and zooming was also a problem. Children (and again, many adults) have trouble using the three-button mouse we currently use for zooming. Small hands can't access all three buttons easily, and few people can remember which button zooms in which direction. To this end, we are developing local tools with animated icons to control panning and zooming which will allow Pad++ to be controlled with a one button mouse.

CONCLUSION

Our preliminary results show that local tools are a promising new way to interact with computer applications. While we are improving the details of our implementation, a wide range of users, including very young children were able to use local tools effectively and reported enjoying them.

In addition to improving the existing local tools, we are working on expanding the tool set and organization. We are planning on providing multiple groups of tools that can easily be accessed, modified, and shared with other users.

ACKNOWLEDGEMENTS

We wish to thank all the children in Albuquerque, NM and at CHIKids who supported our efforts in testing local tools. In addition, we wish to thank Jon Meyer, one of our collaborators at the New York University Media Research Laboratory for discussions about local tools, while he independently did related work on Etcha-Pad.

This work is supported in part by ARPA contract #N66001-94-C-6039.

REFERENCES

[1] Benjamin B. Bederson, James D. Hollan, Ken Perlin, Jonathan Meyer, David Bacon, George Furnas. *A Zoomable Graphical Sketchpad For Exploring Alternate Interface Physics*, Journal of Visual Languages and Computing, 1996, Vol. 7, 3-31.

[2] Benjamin B. Bederson and James D. Hollan, *Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics*, Proceedings of ACM Symposium on User Interface Software and Technology (UIST'94), 17-26.

[3] Benjamin B. Bederson, Larry Stead, and James D. Hollan, *Pad++: Advances in Multiscale Interfaces*, Proceedings of ACM SIGCHI Conference (CHI'94), 315-316.

[4] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. *Toolglass and Magic Lenses: The See-Through Interface*, Proceedings of 1993 ACM SIGGRAPH Conference, 73-80.

[5] Jonathan Meyer. *Etcha-Pad*, Proceedings of ACM SIGCHI Conference Companion (CHI'96), 195-196.

[6] Peter Lucas, Steven F. Roth, Cristina C. Gomberg. *Visage: Dynamic Information Exploration*, Proceedings of ACM SIGCHI Conference Companion (CHI'96), 19-20.

Local Tools: An Alternative to Tool Palettes

Benjamin B. Bederson, James D. Hollan, Allison Druin, Jason Stewart, David Rogers, David Proft
Computer Science Department
University of New Mexico
Albuquerque, NM 87131
([*bederson, hollan, allisond, jasons, drogers, proft*]*@cs.unm.edu*)
http://www.cs.unm.edu/pad++

KEYWORDS
Interactive user interfaces, multiscale zoomable interfaces, information visualization, information physics, local tools.

ABSTRACT
We describe local tools, a general interaction technique that replaces traditional tool palettes. A collection of tools sit on the worksurface along with the data. Each tool can be picked up (where it replaces the cursor), used, and then put down anywhere on the worksurface. There is a toolbox for organizing the tools. These local tools were implemented in Pad++ as part of KidPad, an application for children.

INTRODUCTION

Sitting at a desk, many people find that they work with several tools simultaneously. Perhaps they have a pencil, a red pen, a stapler, or some paper clips - all on their worksurface together. This is a very natural way to work, yet most computer interfaces don't support this style of interaction. Rather, traditional computer tool palettes allow only a single tool to be active at a time. In the real world, this would be equivalent to being forced to put away every tool before another could be used.

In the Pad++ research group at the University of New Mexico, we have been experimenting with an alternative style of interaction we call *local tools*. Motivated by the above scenario, we allow the user to place several tools directly on the work surface, and then pick them up and use them. When a tool is held, it becomes the cursor and can be used just like a regular tool. But these tools can easily be put down anywhere on the work surface, and other tools can be picked up in their place.

One of the driving reasons behind this development of local tools has been our work in developing a zooming application for children using Pad++. We tested the standard tool palette interface in Pad++ with computer-novice fourth graders. What we found was that children had a difficult time with the interface. Despite the fact that the interface used *direct manipulation*, it wasn't direct enough. Tool palettes are hard to use. The user must first find the tool palette (sometimes having to access it from a menu). Then she must press the correct button resulting in the cursor changing - which is quite confusing to the uninitiated computer user. Finally, she must select various attributes of the tool, such as color and width. A local tool can embody all of these characteristics at once. Local tools remain on the surface and can be picked up and used with all of their attributes, potentially reducing cognitive load.

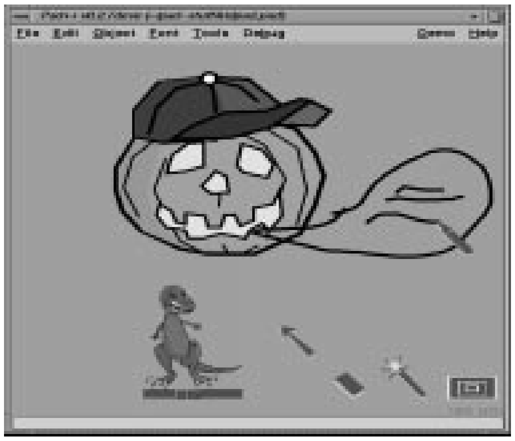

Figure 1: Screen snapshot from KidPad showing use of local tools.

We are building these local tools within Pad++, a zoomable environment [1][2][3]. Pad++ provides a huge worksurface where graphical objects can be put on the surface at any position and at any size. The user can navigate through this planar space by panning and zooming. While local tools could be implemented in a non-zooming environment, there are some drawbacks because the local tools sit on the worksurface taking up valuable screen real estate. In a zoomable environment, however, the tools can easily be made as large or small as desired. In addition, they can be pushed off the screen and then easily brought back with a special *toolbox* (see below).

Developments similar to local tools have recently been described by several research groups. Bier et. al. [4] and our own research group [1] have recently discussed the

Published in ACM UIST'96, pp 169-170.