Pad++: A Zoomable Graphical Interface

Benjamin B. Bederson
Bell Communications Research
445 South Street
Morristown, NJ 07960
(bederson@bellcore.com)

James D. Hollan
Computer Science Department
University of New Mexico
Albuquerque, NM 87131
(hollan@cs.unm.edu)

KEYWORDS
Interactive user interfaces, multiscale interfaces, multimedia authoring, information navigation, hypertext, information visualization.

ABSTRACT
We have implemented an infinite resolution multimedia sketchpad as a base for exploring a *stream-of-consciousness model of computation* where information creating, sharing and retrieval becomes so intuitive that the interface becomes invisible. Motivation to pursue this came from work on Pad [4], which can be thought of as a kind of traditional sketchpad or windows environment in the sense that it is a general-purpose substrate for visualizing two dimensional graphics and text. But Pad also supports the radical notion of being infinite in extent and resolution.

We implemented Pad++ to explore smooth zooming for navigation and to serve as a platform for multimedia authoring and information visualization. The ability to work with very large datasets has been a primary design consideration in the development of Pad++.

INTRODUCTION
The motivation for our work on multiscale interfaces is to move beyond windows, icons, menus, and pointers to explore interfaces that more effectively support navigation through complex sets of multimedia information [1, 2, 5]. One premise of our work is the belief that navigation in information spaces is best supported by tapping into our natural spatial and geographic ways of thinking [3].

Information can be entered interactively by the user or dynamically by a program on Pad++. It can be placed at any location and at arbitrary scale. Navigation is performed by zooming in and out and moving around the surface. Our metaphor for searching for information is that all the information is there and to see more detail, you just have to *take a closer look*. Using Pad is meant to feel like you have a gigantic piece of paper that you can look at with a microscope with arbitrary resolving power and smooth zooming.

RECENT ADVANCES

Our focus in this implementation has been to provide smooth zooming in a system that works with very large datasets. The nature of the Pad++ interface requires consistent high frame-rate interactions, even as the dataspace becomes large and the scene gets complicated. In many applications, speed is important, but not critical to functionality. In Pad++, however, the interface paradigm is inherently based on interaction. The searching strategy is to visually explore the dataspace, so it is essential that interactive frame rates be maintained.

We implemented Pad++ in C++ on Silicon Graphics computers using SGI's graphics language facilities (the GL) to provide smooth zooming. We wrote a standard set of drawing widgets to enter text, lines, squiggles, etc., to load files and images, and a standard set of operators such as grouping, copying, deleting, etc. These give a multimedia author the basic tools for creating a dataspace.

In order to keep the animation frame-rate up as the dataspace size and complexity increases, we implemented several standard efficiency methods, which taken together create a powerful system. We are able to load in over 600,000 objects and maintain interactive rates. Briefly, these methods are:

- **Spatial Indexing:** Create a hierarchy of objects based on bounding boxes to quickly index to visible objects.
- **Spatial Level-Of-Detail:** Render only the detail needed, do not spend time rendering what can not be seen.
- **Clipping:** Only render the portions of objects that are actually visible.
- **Refinement:** Render fast with low resolution while navigating and refine the image when still.
- **Adaptive Render Scheduling:** Keep the zooming rate constant even as the frame rate changes.

One challenge in navigating through any large dataspace is maintaining a sense of relationship between what you are looking at and where it is with respect to the rest of the data. The rough animation or jumpy zooming as implemented in the original Pad [4] can be disorienting and thus not provide the most effective support for the cognitive and perceptual processing required for interactive information visualization and navigation. We implemented smooth zooming where

Published in CHI '94: short paper category

pressing the middle button zooms in and pressing the right button zooms out (the left button is a context-dependent button). Pressing either of the zoom buttons once and holding it initiates the zooming which starts slowly and accelerates according to a profile specified by the maximum zooming speed and how long it takes to reach that speed. Holding the middle and right buttons permit panning.

An important interactive interface issue when accessing external information sources is how to give the user access to them without incurring substantial start-up costs while the database is parsed and loaded. In Pad++ this is accomplished with *parallel lazy loading*: only load the portion of the database that is visible in the current view. As the user navigates through the database and looks at new areas, those portions of the database are loaded. This lazy loading is done in the background so the user can continue to interact with Pad++, and when the loading is complete, items appear in the appropriate place.

SAMPLE APPLICATION

We built a Pad++ directory browser to explore how smooth zooming and the various efficiency mechanisms help in viewing a large hierarchical database. The Pad++ directory browser provides a graphical interface for accessing the directory structure of a filesystem [Figure 1]. Each directory is represented by a square pink frame, and files are represented by solid squares colored by file type (Figure 1 is B&W). Both directories and files show their filenames as labels when the user is sufficiently close to be able to read them. Each directory has all of its subdirectories and files organized alphabetically inside of it. Searching through the directory structure involves only zooming in and out of the directory tree. Zooming into a file automatically loads the text inside the colored square and it can then be edited and annotated.

Figure 1: Directory browser snapshot

We are able to load in a directory tree with over 600,000 objects, and maintain interactive animation rates of about 10 frames per second. Spatial indexing allows us to maintain a very large database while keeping the search time fast, since we render only the visible portion. While navigating with Pad++, very small objects are not drawn and larger ones are drawn with reduced resolution. The reduced resolution objects are refined when the user stops changing the view.

CONCLUSION

We implemented Pad++, a multimedia multiscale hierarchical sketchpad, focusing on efficiency and expandability. By implementing several efficiency mechanisms which act in concert, we are able to maintain high frame-rate interaction with very large databases.

We are currently, in collaboration with NYU, beginning to design an end user language for application authoring. In addition, we also are exploring several application domains, such as history-enriched digital objects, as part of the continued development of Pad++.

ACKNOWLEDGEMENTS

We would like to acknowledge other members of the Computer Graphics and Interactive Media Research Group at Bellcore for many discussions shared during our search for the best cheeseburger. We also would like to thank Ken Perlin and his students, David Fox and Matthew Fuchs, at NYU for enjoyable discussions and for seeding out interest in multiscale interfaces.

REFERENCES

[1] Benjamin B. Bederson, Larry Stead, and James D. Hollan, Pad++: Advances in MultiScale Interfaces, *Proceedings of ACM Human Factors in Computing Systems Conference (CHI '94)*.

[2] Stuart K. Card, George G. Robertson, and Jock D. Mackinlay. The Information Visualizer, an Information Workspace, *Proceedings of ACM Human Factors in Computing Systems Conference (CHI '91)*, 181-188.

[3] William C. Donelson. Spatial Management of Information, *Proceedings of 1978 ACM SIGGRAPH Conference*, 203-209.

[4] Kim M. Fairchild, Steven E. Poltrock, and George W. Furnas. SemNet: Three-Dimensional Graphic Representations of Large Knowledge Bases, in *Cognitive Science and its Applications for Human-Computer Interaction*, Lawrence Erlbaum Associates, 1988.

[5] George W. Furnas. The FISHEYE View: a New Look at Structured Files, Bell Laboratories Technical Report, 11221-22, 1982.

[6] George Lakoff and Mark Johnson. Metaphors We Live By. University of Chicago Press, 1980.

[7] Ken Perlin and David Fox. Pad: An Alternative Approach to the Computer Interface, *Proceedings of 1993 ACM SIGGRAPH Conference*, 57-64.

[8] Ivan E. Sutherland. Sketchpad: A man-machine graphical communications systems, *Proceedings of the Spring Joint Computer Conference*, 1963, 329-346, Baltimore, MD: Spartan Books.