Journal of Visual Languages and Computing (1996) **7**, 3–31

Pad++: A Zoomable Graphical Sketchpad For Exploring Alternate Interface Physics

Benjamin B. Bederson,[*] James D. Hollan,[*] Ken Perlin,[†] Jonathan Meyer,[†] David Bacon[†] and George Furnas[‡]

[*]Computer Science Department, University of New Mexico, Albuquerque, NM 87131, U.S.A., [†]Media Research Laboratory, Computer Science Department, New York University, NY 10003, U.S.A., [‡]Bell Communications Research, 445 South Street, Morristown, NJ 07960, U.S.A.

Received 7 March 1995 and accepted 29 September 1995

We describe Pad++, a zoomable graphical sketchpad that we are exploring as an alternative to traditional window and icon-based interfaces. We discuss the motivation for Pad++, describe the implementation and present prototype applications. In addition, we introduce an informational physics strategy for interface design and briefly contrast it with current design strategies. We envision a rich world of dynamic persistent informational entities that operate according to multiple physics specifically designed to provide cognitively facile access and serve as the basis for the design of new computationally-based work materials.

1. Introduction

Imagine a computer screen made of a sheet of a miraculous new material that is stretchable like rubber but continues to display a crisp computer image, no matter what the sheet's size. Imagine that this sheet is very elastic and can stretch orders of magnitude more than rubber. Further, imagine that vast quantities of information are represented on the sheet, organized at different places and sizes. Everything you do on the computer is on this sheet. To access a piece of information you just stretch to the right part and there it is.

Imagine further that special lenses come with this sheet that let you look onto one part of the sheet while you have stretched another part. With these lenses, you can see and interact with many different pieces of data at the same time that would ordinarily be quite far apart. In addition, these lenses can filter the data in any way you would like, showing different representations of the same underlying data. The lenses can even filter out some of the data so that only relevant portions of the data appear.

Imagine also new stretching mechanisms that provide alternatives to scaling objects purely geometrically. For example, instead of representing a page of text so small that it is unreadable, it might make more sense to present an abstraction of the text, perhaps so that just a title that is readable. Similarly, when stretching out a spreadsheet,

[*] *(bederson, hollan)@ cs.umn.edu,* [†] *(perlin, meyer bacon)@ play.cs.nyu.edu,* [‡]*gwf@bellcore.com*
URL:http://www.cs.unm.edu/pad++

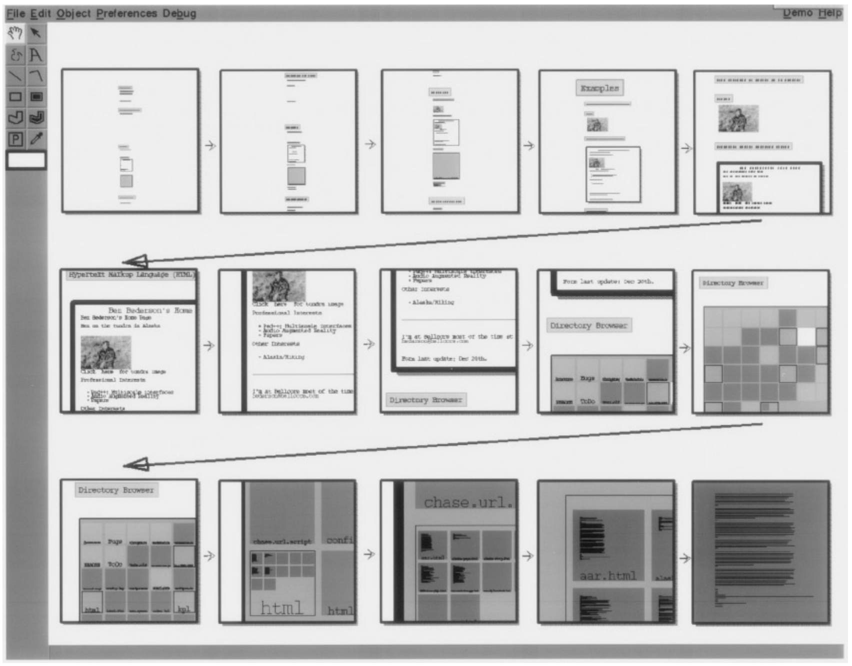

Figure 1. A sequence of views as we zoom into some data

instead of showing huge numbers it might make more sense to show the computations from which the numbers were derived or a history of interaction with them.

The beginnings of an interface like this sheet exists today in a program we call Pad++. We don't really stretch a huge rubber-like sheet, but we simulate it by *zooming* into the data. We use what we call *portals* to simulate lenses, and a notion we call *semantic zooming* to scale data in non-geometric ways. The user controls where they look on this vast data surface by panning and zooming. Portals are objects on the Pad++ data surface that can see anywhere on the surface, as well as filter data to represent it differently than it normally appears.

Panning and zooming allow navigation through a large information space via direct manipulation. By tapping into people's natural spatial abilities, we hope to increase users' intuitive access to information. Conventional computer search techniques are also provided in Pad++, bridging traditional and new interface metaphors. Figure 1 depicts a sequence of views as we pan and zoom into some data.

1.1. Motivation

If interface designers are to move beyond windows, icons, menus and pointers to explore a larger space of interface possibilities, additional ways of thinking about interfaces that go beyond the desktop metaphor are required.

There are myriad benefits associated with metaphor-based approaches, but they also orient designers to employ computation primarily to mimic mechanisms of older media. While there are important cognitive, cultural and engineering reasons to exploit earlier successful representations, this approach has the potential of underutilizing the mechanisms of new media.

For the last few years we have been exploring a different strategy [21] for interface design to help focus on novel mechanisms enabled by computation rather than on mimicking mechanisms of older media. Informally, the strategy consists of viewing interface design as the development of a physics of appearance and behavior for collections of informational objects.

For example, an effective informational physics might arrange for an object's representation to be a natural by-product of normal activity. This is similar to the physics of certain materials that evidence the wear associated with use. Such wear records a history of use and at times this can influence future use in positive ways. Used books crack open at frequently referenced places. It is common for recently consulted papers to be at the tops of piles on our desks. Usage dog-ears the corners and stains the surface of index cards and catalogs. All these wear marks provide representational cues as a natural product of doing, but the physics of materials limit what can be recorded and the ways it can influence future use.

Following an informational physics strategy has led us to explore history-enriched digital objects [18, 19]. Recording on objects (e.g. reports, forms, source-code, manual pages, email, spreadsheets) the interaction events that comprise their use makes it possible on future occasions, when the objects are used again, to display graphical abstractions of the accrued histories as parts of the objects themselves. For example, we depict the copy history on source code. This allows a developer to see that a particular section of code has been copied and perhaps be led to correct a bug not only in the piece of code being viewed but also in the code from which it was derived.

This informational physics strategy has also lead us to explore new physics for interacting with graphical data. As part of that exploration we have formed a research consortium to design a successor to Pad [25]. This new system, Pad++, serves as a substrate for exploration of novel interfaces for information visualization and browsing in complex, information-intensive domains. The system is being designed to operate on platforms ranging from high-end graphics workstations to PDAs (Personal Digital Assistants) and interactive set-top cable boxes. Here we describe the motivation behind the Pad++ development, report the status of the current implementation and present initial prototype applications.

Today, there is much more information available than we can access readily and effectively. The situation is further complicated by the fact that we are on the threshold of a vast increase in the availability of information because of new network and computational technologies. Paradoxically, while we continuously process massive amounts of perceptual data as we experience the world, we have perceptual access to very little of the information that resides within our computing systems or that is reachable via network connections. In addition, this information, unlike the world around is, is rarely presented in ways that reflect either its rich structure or dynamic character.

We envision a much richer world of dynamic persistent informational entities that operate according to multiple physics specifically designed to provide cognitively facile access. These physics need to be designed to exploit semantic relationships explicit and implicit in information-intensive tasks and in our interaction with these new kinds of computationally-based work materials.

One physics central to Pad++ supports viewing information at multiple scales and attempts to tap into our natural spatial ways of thinking. We address the information presentation problem of how to provide effective access to a large structure of information on a much smaller display. Furnas [15] explored degree of interest functions to determine the information visible at various distances from a central focal area. There is much to recommend the general approach of providing a central focus area of detail surrounded by a periphery that places the detail in a larger context.

With Pad++ we have moved beyond the simple binary choice of presenting or eliding particular information. We can also determine the scale of the information and, perhaps most importantly, the details of how it is rendered can be based on various semantic and task considerations that we describe below. This provides semantic task-based filtering of information that is similar to the early work at MCC on lens-based filtering of a knowledge base using HITS [20] and the recent work of moveable filters at Xerox [4] [30].

The ability to make it easier and more intuitive to find specific information in large dataspaces is one of the central motivations behind Pad++. The traditional approach is to filter or recommend a subset of the data, hopefully producing a small enough dataset for the user to navigate effectively. Pad++ is complementary to these filtering approaches in that it promises to provide a useful substrate to *structure* information.

2. Description

Pad++ is a general-purpose substrate for creating and interacting with structured information based on a zoomable interface. It adds scale as a first class parameter to all items, as well as various mechanisms for navigating through a multiscale space. It has several efficiency mechanisms which help maintain interactive frame-rates with large and complicated graphical scenes.

While Pad++ is not an application itself, it directly supports creation and manipulation of multiscale graphical objects, and navigation through spaces of these objects. It is implemented as a widget in Tcl/Tk [24] (described in a later section) which provides an interpreted scripting language for creating zoomable applications. The standard objects that pad++ supports are colored text, graphics, images, portals and hypertext markup language (HTML). Standard input widgets (buttons, sliders, etc.) are supplied as extensions.

One focus in the current implementation has been to provide smooth zooming within very large graphical 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 interactive. One important searching strategy is to visually explore the dataspace while zooming through it, so it is essential that interactive frame rates be maintained.

A second focus has been to design Pad++ to make it relatively easy for third parties to build applications using it. To that end, we have made a clear division between what we call the 'substrate' and applications. The substrate, written in C++, is part of every release and has a well-defined API. It has been written with care to ensure efficiency and generality. It is connected to a scripting language (currently Tcl, but we are exploring alternatives) that provides a fairly high-level interface to the complex graphics and interactions available. While the scripting language runs quite slowly, it is used as a glue language for creating interfaces and putting them together. The actual interaction and rendering is performed by the C++ substrate. This approach allows people to develop applications for Pad++ while avoiding the complexities inherent in this type of system. (See the Implementation section for more information on this.)

2.1. PadDraw: A Sample Application

PadDraw is a sample drawing application built on top of Pad++. It supports interactive drawing and manipulation of objects as well as loading of predefined or programmatically created objects. This application is written entirely in Tcl (the scripting language) and was used to produce all the figures depicted in this paper. The tools, such as navigation aids, hyperlinks and the outline browser, that we discuss later, are part of this application.

The basic user interface for navigating in PadDraw uses a three button mouse. The left button is mode dependent and lets users select and move objects, draw graphical objects, follow hyperlinks, etc. The middle button zooms in and the right button zooms out. Zooming is always centered on the cursor, so moving the mouse while zooming lets the user dynamically control which point they are zooming around.

PadDraw has a primitive Graphical User Interface (GUI) builder that is in progress. Among other things, it allows the creation of active objects. Active objects can animate the view to other locations (a kind of hyperlink) or move other objects around on the surface.

2.1.1. Navigation

Easily finding information on the Pad++ surface is obviously very important since intuitive navigation through large dataspaces is one of its primary motivations. Pad++ supports visual searching with direct manipulation panning and zooming in addition to traditional mechanisms, such as content-based search.

Some applications animate the view to a certain piece of data. These animations interpolate in pan and zoom to bring the view to the specified location. If the end point is further than one screen width away from the starting point, the animation zooms out to a point midway between the starting and ending points, far enough out so that both points are visible. The animation then smoothly zooms in to the destination. This gives both a sense of context to the viewer as well as speeding up the animation since most of the panning is performed when zoomed out which covers much more ground than panning while zoomed in. See the section on Space-Scale Diagrams for more detail on the surprisingly complex topic of multiscale navigation.

Content-based search mechanisms support search for text and object names. Entering text in a search menu results in a list of all of the objects that contain that text. Clicking on an element of this list produces an automatic animation to that object. The search also highlights objects on the data surface that match the search specification with special markers (currently a bright yellow outline) that remain visible no matter how far you zoom out. Even though the object may be so small as to be invisible, its marker will still be visible. This is a simple example of task-based semantic zooming. See Figure 2 for a depiction of the content-based search mechanism.

We have also implemented visual bookmarks as another navigational aid. Users can

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.