SFDC 1011

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

Chart

Claim Chart of Bederson et al., *Pad++: A Zoomable Graphical Sketchpad For Exploring Alternate Interface Physics*, Journal of Visual Languages and Computing (1996) ("Bederson I")

and

Bederson et al., *A Zooming Web Browser*, in Proceedings of SPIE Conference on Multimedia Computing and Networking, 1996 ("Bederson II") (collectively "Pad++")

as prior art to

Asserted Claims of U.S. Patent No. 7,356,482 ("the '482 Patent")

'482 Patent	PAD++
Claim 1	
A system for providing a dynamically generated application having one or more functions and one or more user interface elements; comprising:	To the extent that this preamble is construed to be limiting, PAD++ discloses a system for providing a dynamically generated application having one or more functions and one or more user interface element. *See, e.g.*: BEDERSON I, at 4: "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 ." BEDERSON I, at 5-6: "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 .

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

‘482 Patent	PAD++
	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 thirparties 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 witcare 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.)” BEDERSON I, at 7: “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 .” BEDERSON I, at 11: “For example , we implemented a digital

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

'482 Patent	PAD++
	clock that at normal size shows the hours and minutes . When zooming in , instead of making the text very large , it shows the seconds , and then eventually the date as well . Similarly , zooming out shows just the hour . An analog clock (implemented as a lens that can be positioned over a digital clock) is similar—it does not show the second hand or the minute markings when zoomed out." BEDERSON I, at 11: "We are exploring several different types of interactive visualizations within Pad++, some of which are described briefly here . Each takes advantage of the variable resolution available for both representation and interaction . Layout of graphical objects within a multi-resolution space is an interesting problem , and is quite different than traditional fixed-resolution layout . Deciding how to visually represent an arbitrary graph on a non-zoomable surface is extremely difficult . Often it is impossible to position all objects near logically related objects . In addition , representing the links between objects often requires overlapping or crossing edges . Even laying out a tree is difficult because , generally speaking , there are an exponential number of children that will not fit in a fixed size space . Traditional layout techniques use sophisticated iterative , adaptive algorithms for laying out general graphs , and still result in graphs that are hard to understand . Large trees are often represented hierarchically with one sub-tree depicted by a single box that references another tree . Using an interactive zoomable surface , however , allows very different methods of visually representing large data structures . The fact that there is always more room to put information 'between the cracks' gives many more options . Pad++ is particularly well suited to visualizing hierarchical data because information that is deeper in the hierarchy can be made smaller . Accessing this information is accomplished by zooming ." BEDERSON I, at 12: "We built a zoomable directory browser as another exploration of multiscale layout . The directory browser provides a graphical interface for accessing the directory structure of a filesystem (see Figure 6). Each directory is represented by a folder icon and files are represented by solid squares colored by file type . 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 it . Searching through the directory structure can be done by zooming in and out of the directory tree , or by using the content based search mechanisms described above . Zooming into a file automatically loads its text or image inside the

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

‘482 Patent	PAD++
	colored square and it can then be annotated . At any particular view , typically three levels of the hierarchy are visible .” BEDERSON II, at 3: “We are exploring dynamic multiscale techniques to support focus and context during navigation of large information spaces. To accomplish this we are building a zoomable web browser using Pad++, a substrate for building multiscale dynamic user interfaces [2][3][27][28]. Pad++ provides an extensive graphical workspace where dynamic objects can be placed at any position and at any scale. Pad++ supports panning and zooming. Zooming can involve simple geometric scaling or what we term *semantic zooming*, in which rendering of objects can vary based on factors in addition to scale, such as context of the task or complexity of the information being displayed. Pad++ is built as a widget for Tcl/Tk, a scripting language and user-interface library [26][33]. Pad++ allows WWW pages to remain visible at varying scales while they are not specifically being visited, so the viewer may examine many pages at once. In addition, Pad++ allows the user to zoom in and out of pages, enabling explicit control of how much context is viewed at any time. To orient themselves, users can simply zoom back to view a number of web pages. To get more detailed views of a particular page they can zoom in. We think this variable scale contextual display of web pages can provide important support for navigation. We are currently exploring a tree layout system that permits users to dynamically add to and reorganize a tree of web pages. Using our Pad++ web browser, users navigate a space filled with familiar objects, not iconified representations of those objects. Our dynamic Pad++ tree browser combines a basic focus-driven layout with automatic zooming and panning to support navigation. The software allows the user to select a focus page. That selection animates the page to occupy a larger section of the display. Pages farther from the focus page get increasingly smaller, resulting in a graphical fisheye view [30]. See Figures 1 and 2 for snapshots of the Pad++ web browser during reorganization.” BEDERSON II, at 7: “Currently there are a number of efforts to create more interactive WWW documents. The primary approach is to write code in a programming language, instead of HTML, that can be downloaded into a browser equipped to interpret the language. Sun’s Hot-Java project uses the Java language [18], Cygnus Support’s GNU Remote Operations Web (GROW) proposes to use GNU’s Guile extension language [17], and Microsoft’s Blackbird will use dynamically loadable object files

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

DOCKET ALARM

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.