A Zooming Web Browser

Benjamin B. Bederson, James D. Hollan,
Jason Stewart, David Rogers, Allison Druin, David Vick
Computer Science Department
University of New Mexico
Albuquerque, NM 87131
{bederson, hollan, jasons, drogers, allisond, dvick}@cs.unm.edu
http://www.cs.unm.edu/pad++

ABSTRACT

The World Wide Web (WWW) is becoming increasingly important for business, education, and entertainment. Popular web browsers make access to Internet information resources relatively easy for novice users. Simply by clicking on a link, a new page of information replaces the current one on the screen. Unfortunately however, after following a number of links, people can have difficulty remembering where they've been and navigating links they have followed. As one's collection of web pages grows and as more information of interest populates the web, effective navigation becomes an issue of fundamental importance.

We are developing a prototype zooming browser to explore alternative mechanisms for navigating the WWW. Instead of having a single page visible at a time, multiple pages and the links between them are depicted on a large zoomable information surface. Pages are scaled so that the page in focus is clearly readable with connected pages shown at smaller scales to provide context. As a link is followed the new page becomes the focus and existing pages are dynamically repositioned and scaled. Layout changes are animated so that the focus page moves smoothly to the center of the display surface while contextual information provided by linked pages scales down.

While our browser supports multiscale representations of existing HTML pages, we have also extended HTML to support multiscale layout *within* a page. This extension, *Multi-Scale Markup Language* (MSML), is at an early stage of development. It currently supports inclusion within a page of variable-sized dynamic objects, graphics, and other interface mechanisms from our underlying Pad++ substrate. This provides sophisticated client-side interactions, permits annotations to be added to pages, and allows page constituents to be used as independent graphical objects.

In this paper, we describe our prototype web browser and authoring facilities. We show how simple extensions to HTML can support sophisticated client-side interactions. Finally, we discuss the results of preliminary user-interface testing and evaluation.

Keywords: world-wide web, browser, information navigation, zooming, information visualization, multiscale information, animated user interface, Pad++.

1. INTRODUCTION

In 1945 Vannevar Bush [8] envisioned "a future device for individual use, which is a sort of mechanized private file and library." He termed this device a *memex* and proposed a form of associative indexing in which arbitrary pieces of information could be linked together such that "when one of these items is in view, the other can be instantly recalled by tapping a button." He further conjectured that "wholly new forms of encyclopedias will appear, ready made with a mesh of associative trails running through them, ready to be dropped into the memex and there amplified." Today, fifty years later, we have the World Wide Web and a memex in the form of web browsers. See [4] for an overview of the WWW.

The increasing number of users and the ever-growing quantity of information available on the web present challenging interface and navigation problems. There are a variety of human factors [19] issues that need to be addressed. A larger number of users means that people with diverse talents, interests, and experiences will be on-line via the web. Many will be novices with little prior experience with computers. A simple click of the mouse can bring a user from their friend's home page to unknown destinations across the world. Traditionally, following a cross reference meant shuffling across the library to find another volume. While time-consuming, this reinforced the transition that was taking place. The difficulties that novice users confront can be instructional to developers. While experts may not have as much difficulty, they experience the same cognitive burdens, and may just have a higher threshold before they experience similar difficulties.

While the immediacy of traversing information links offers many advantages, it can also make it difficult to maintain an intuitive sense of where one is, and how one got there - leading to the frequently described sense of being *lost*. This is a classic problem of hypertext systems. Part of the problem can be attributed to windows-based interfaces. Current window systems don't readily support showing more than a few pages at a time. In addition, each page is usually in a separate window with no depiction of relationships to other windows. Popular WWW browsers, like other applications built according to current tiled or overlapping windows philosophies, also have this same problem, although they do offer limited methods to aid navigation by keeping track of interesting sites - usually in hierarchical sets of *hotlists* or *bookmarks*.

Several groups have proposed alternatives and extensions to browsers to address some aspects of this problem. Oostendorp describes the PAINT system (Personalized Adaptive Internet Navigation Tool) [25]. It provides an interface for accessing hierarchies of bookmarks in a style similar to the NEXTStep interface. WebMap is a browser extension that shows a graphical relationship between web pages [11]. Each page is represented by a small circle that can be selected to display the actual page. The links between pages are colored to indicate information about the links, such as whether it is a link to a different server or whether the destination page has already been read. These graphs may be saved and used by others.

While the web is inherently cyclic, it is easier to visualize hierarchies, and so many web visualizations are based on hierarchies extracted from the graph of the web. Some interesting work focuses on alternative visualizations [24]. Furnas [14] shows how *multitrees* can be used to represent a collection of hierarchies sharing parts of the underlying data. One application of multitrees is visualization of bookmarks from multiple individuals[34]. Furnas [16] also describes a framework for characterizing how different structures influence effective view traversal, the mechanical process of moving between information items, and view navigation, finding good paths to information items.

Another approach to visualizing large information spaces that can be applied to web browsing and navigation involves techniques to show detail at particular nodes while maintaining context. One general approach, fisheye views [13], has been extended with graphics [30], three dimensions [9][22], hyperbolic representations[20], animation [10], and zooming [2][3][28]. Other techniques include exploiting a large virtual space [12], using lenses or filters [5][23][31], and visualizing two dimensional layouts [1][21].

In addition to the difficulty of finding information, it becomes ever more important to tailor information for one's own needs. Also rather than searching oneself can be sensible to go by other's recommendations. This is the basis for commercial services such as Yahoo [36], and follows the often effective strategy of exploiting recommendations from those one knows and trusts [35].

Annotations are another important information tailoring facility. Annotations are personal markings that can be used to highlight and comment on information for oneself and others. One interesting approach to annotation on the web separates the annotations from the original documents and stores them in a special annotation server [29]. Used with an enhanced browser, displaying a new page automatically brings in the annotations of others and integrates them into the page.

In the sections that follow, we describe our zooming web browser and the attempt to use animation and multiscale representation of context to support more effective web navigation. In addition to visualization of standard HTML pages, we introduce extensions to HTML that allow more sophisticated presentations and client-side interactions. We demonstrate the beginnings of direct manipulation graphical authoring tools and show how annotation can be supported as a form of authoring. Finally, we present the results of initial user testing and envision a scenario for future web use in the classroom.

2. A ZOOMING WEB BROWSER

Navigating the WWW presents a struggle between focus and context. As one browses or searches the web the need for detailed views of specific items conflicts with the need to maintain a global view of context and history of traversal. This struggle is made more difficult by the haphazard organization of the WWW. Information items closely related by links are not necessarily closely related by content nor in terms of the user's information needs. At times, one seems more likely to find something of interest when not looking for it than when specifically searching. This serendipitous contact with information, though at times frustrating, can also be an advantage. The challenge is how to best support both incidental and intentional access while organizing useful information so that it can be effectively retrieved again in the future.

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.

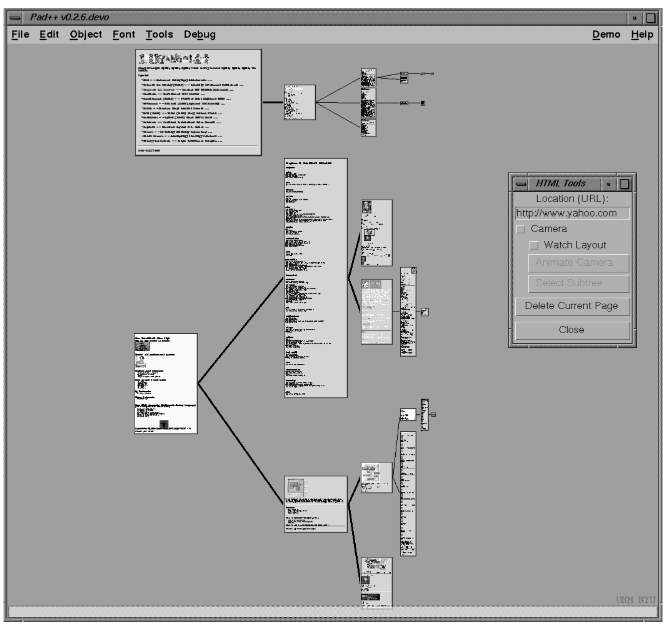

Figure 1: Snapshot of Pad++ Web Browser.

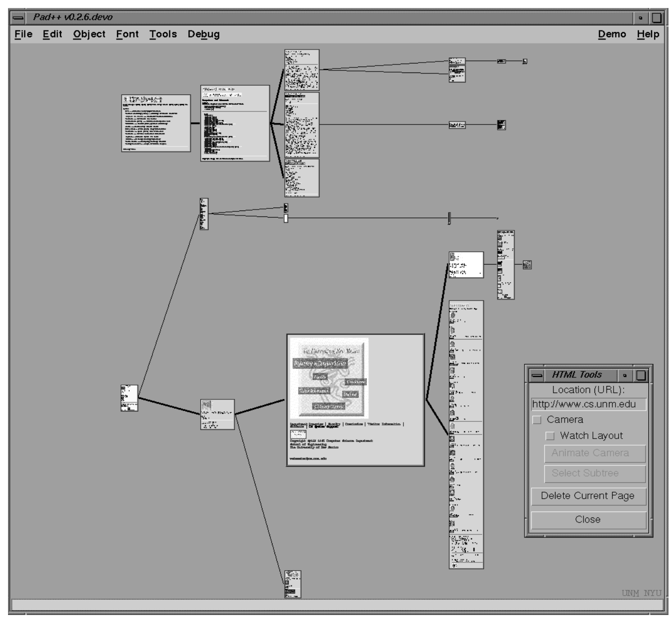

Figure 2: Another view of same web pages.

The Pad++ WWW browser combines Pad++'s interactive multiscale display with dynamic objects that can restructure themselves in response to user actions. Clicking on a link brings up a new page, adds it to the tree of pages, and causes the tree to restructure itself. Unlike other web browsers that immediately replace the current page with a new page, the restructuring process is animated so that users can understand how the tree is being reorganized. The animation helps maintain object constancy and the graphical depiction of links highlights relationships between pages. The new page becomes the current focus and is moved to the center of the screen, at a size suited for viewing. The user may designate any existing page to be the current focus by clicking on it.

As in earlier fisheye displays, our basic layout function assigns a *degree-of-interest value* to each node in the tree based on its distance from the focus page. We define the distance to be the shortest path between two pages[13]. This value is then used to determine the size of each node. See [11] for a description of other hierarchical layout techniques not based on fisheye views.

The layout described above provides a sense of context while following links. We have also implemented an alternative *camera mode* of navigation. It shows the web of links on one side of the screen with a zoomed in view of the focus page on the other side of the screen. A camera is depicted along with the web of pages. The camera can be dragged around or automatically animated through the web. The zoomed in view shows the page the camera is currently looking at (Figure 3). This mode also supports automated tours. For example, one type of camera can take you on a tour of all the parts of your saved web pages that have changed since you last looked at them.

We are currently experimenting with more flexible mechanisms for dynamic tree layout and interaction. These include exploring alternative visualizations and better methods for managing and interacting with large dynamic trees. New tree layout methods will work with any kind of item on the Pad++ surface. Thus, in addition to HTML pages, users will be able to create spaces using any Pad++ object, including drawings, interactive maps, and text.

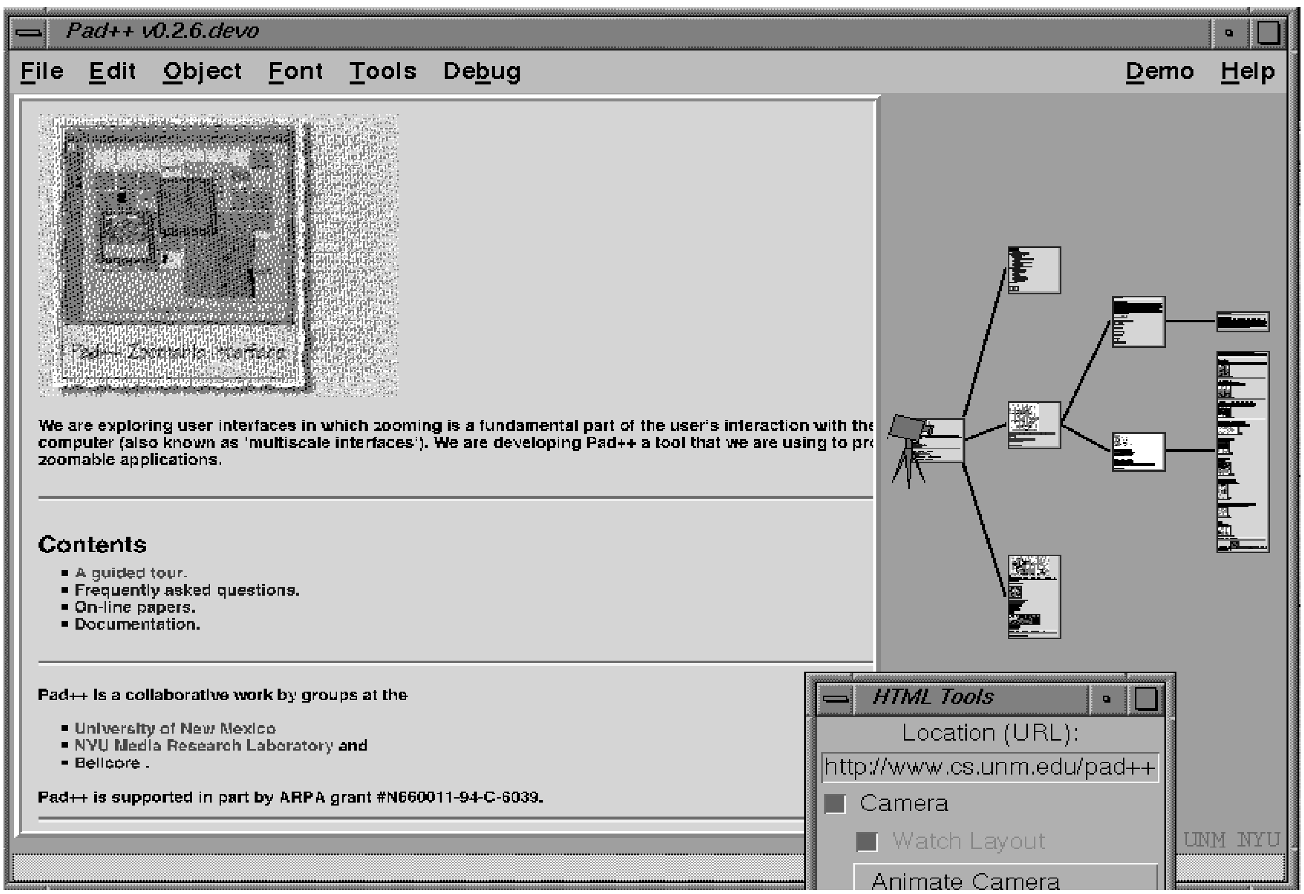

Figure 3: Camera view of web pages.

The new layout code is designed to be hierarchical, so that users may designate subtrees to have different layouts. This allows greater freedom in grouping and display. For example, a certain information tree may contain nodes with subtrees consisting of hundreds or thousands of nodes each. These nodes could exploit a hyperbolic layout to compress the information and the hyperbolic nodes themselves might be layed out radially [20].

Another topic we are exploring involves tradeoffs between maintaining pointers to information on the Web and making copies of the information locally. For example, a user might want to copy items from a remote page to prevent that information from being lost. At other times users may wish to maintain only pointers to information since it is being maintained elsewhere. There are interesting related issues of annotation, that we discuss later, as well as issues of maintaining annotation placement as pages change.

3. AUTHORING WEB PAGES

Thus far, we have discussed visualization and navigation of WWW pages written using standard HTML. We are also exploring extensions to HTML to allow web page authors access to Pad++ multiscale visualization and layout facilities. This extension, *Multi-Scale Markup Language* (MSML), enables users to include arbitrary Pad++ objects in web pages. We have implemented MSML using the HTML *<Meta> tag* and thus added features are invisible to HTML browsers.

Important motivations for MSML are to make WWW pages and their components first-class Pad++ objects and allow authors using MSML to exploit all Pad++ facilities. In addition to adding zooming and other dynamic features to web pages, the goal is permit authors to manipulate and interact with any web page element. Making elements first-class Pad++ objects will result in authors being able to move, scale, delete, add, or modify them. Our approach is similar in spirit to the SELF project [32]. While much remains to be accomplished to support the full informational physics [3] we envision, all of the examples detailed below work in the current Pad++ Web browser.

HTML provides very few basic types. Examples include text, images, bullets items, and horizontal rules. MSML in contrast, supports not only HTML types but also provides access to all the graphical features of Pad++. This gives users a richer toolkit of objects to use when creating documents: text (Postscript Type 1 fonts), lines, rectangles, ovals, lenses (providing filtering and alternative representation), portals (furnishing additional zoomable views of the Pad++ surface), compound objects created from these basic elements, and access to Pad++'s dynamic zooming and panning facilities.

Zooming and Scale

HTML provides header tags, <H1>, <H2>, etc., to indicate degree of importance of sections in WWW documents. However, this mechanism only works with text, not with other media such as images. MSML introduces a method to control the size of all types of objects, including images and graphics. We have used this, for example, to create a multiscale hierarchical outline (see Figure 4). All MSML extensions are written using special Meta-tag keys. This approach makes it clear that the included code is an extension and allows it to simply be ignored by standard HTML-based web browsers.

A portion of the MSML required to create the outline depicted in Figure 4 is given below. The *pad_scale* key takes a single argument that multiplies the current scale and affects the size of all future objects until a */pad_scale* key is seen.

```
<html>
Introduction

<meta pad_scale=0.25>
<ul>
<li>Sistine Chapel
<li>Push the interface metaphor
<li>History

<meta pad_scale=0.25>
<ul>
<li>Ivan Sutherland, SketchPad, 1963
<li>William Donelson, MIT, 1978
<li>George Furnas, Fisheye Views - Bellcore, 1986
<li>Ken Perlin, David Fox, PAD - NYU, 1993
</ul>
<meta /pad_scale>

</ul>
<meta /pad_scale>
...
```

Another example use of MSML involves inclusion of a multiscale state map on a web home page. As the view is zoomed in, first counties, then cities, and then street maps of cities are shown. Finally, even the location of one's home or work could be indicated on the street map. It would even be possible to continue zooming until a floor plan of home or work location becomes visible. See Figures 5-7 for a sequence of snapshots as we zoom into New Mexico. We first see county names and ultimately an Albuquerque street map.

Below is the MSML code that produced the examples in Figure 5-7:

```
<html>
This is the New Mexico Map page
<hr>
<meta pad_tcl={msml_load_tcl http://www.cs.unm.edu/~bederson/pad/county.tcl county}>
```

The map data is stored in a separate code file and is loaded using the `pad_tcl` tag. It passes a Tcl script that uses the MSML library function *msml_load_tcl*. This function takes two arguments: a URL to a Pad++ Tcl script and a tag name to be associated with every object the script creates. It is via this tag that objects are associated with the HTML page.

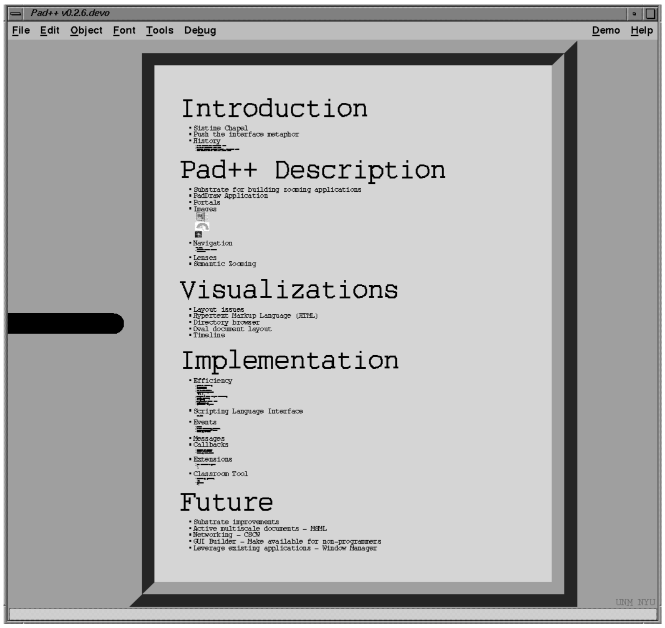

Figure 4: Outline using MSML Scale Tag.

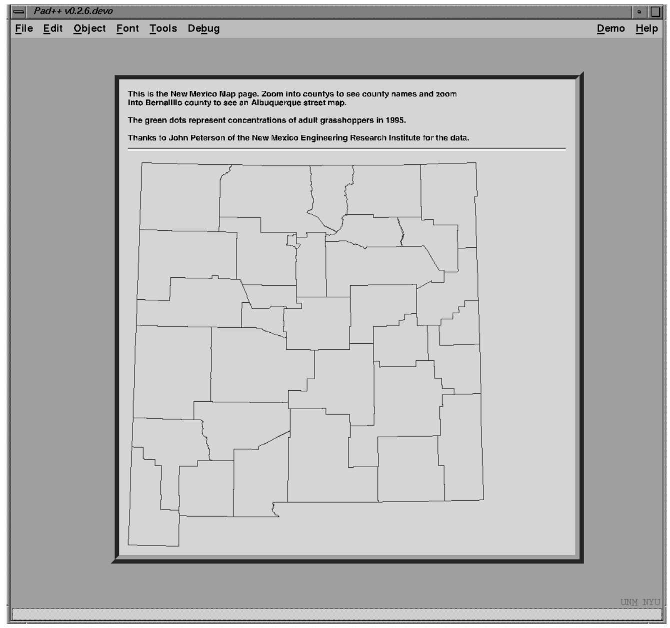

Figure 5: New Mexico county data.

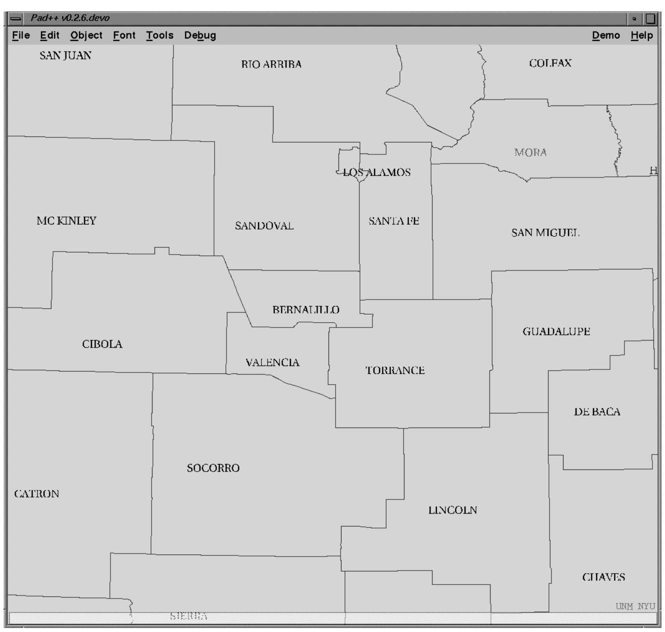

Figure 6: NM zoomed into county names.

Figure 7: NM zoomed into Albuquerque.

The URL specifying `county.tcl` contains Tcl code that creates the county data. The `msml_load_tcl` function inserts it into the Web page and properly scales it. One advantage of MSML is that almost any object on the Pad++ surface can be used as an anchor, not just text and images. So once a user has zoomed into the New Mexico state map and located Albuquerque, the homes of the members of the Pad++ group as well as the Computer Science department itself could be anchors to other WWW pages, or to other points on the map, such as those of collaborators from other parts of the world.

Interaction

Traditionally, browsers come with pre-defined functions and all interactions with a web document are constrained to those functional abilities. Limited animation is possible through techniques such as *server-push* and *client-pull*. Forms, a simple interface built into most clients to collect information and send it to the server for processing, provide constrained GUI-like interactions. However, until recently, nothing supporting more interactive and flexible interfaces has been available.

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 HotJava 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 [6]. By providing the ability to download and run code locally allows complicated animations, for example, to be encoded in a concise and network efficient manner.

The Pad++ WWW browser contains a full Tcl interpreter. MSML provides mechanisms to include Tcl code within the page, instruct the browser to download either scripts or saved Pad++ data files over the WWW, and pass the code to the Tcl interpreter. This, of course, carries with it enormous security risks that we have not yet addressed. In the future we may restrict ourselves to a safe subset of the language as with Java or GROW. In our case this will initially be SafeTcl [7] but may also include support for other languages (e.g. Java).

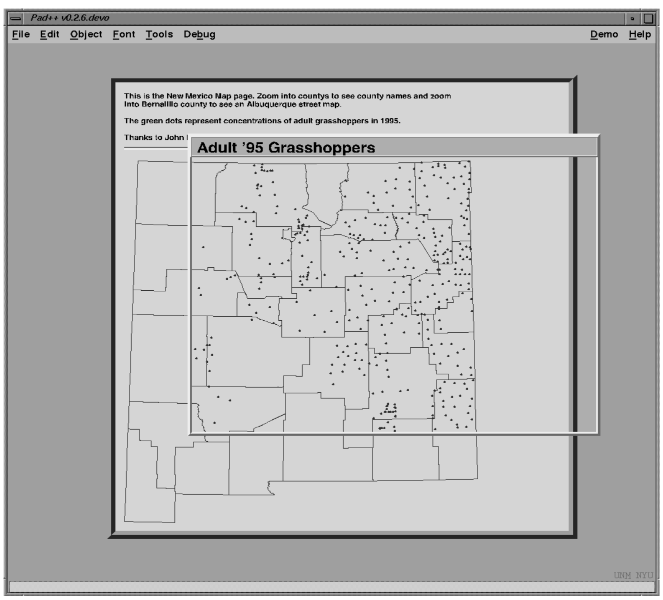

Figure 8: The grasshopper filter applied to the New Mexico County data.

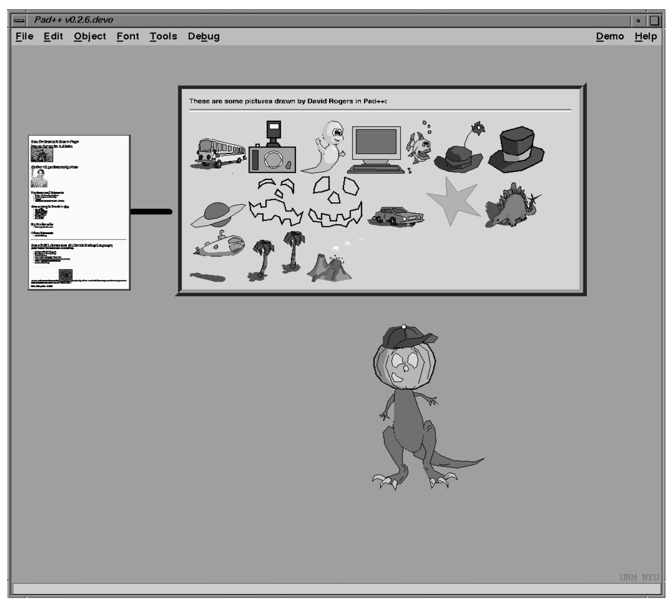

Figure 9: A WWW page containing Pad++ data files with some elements dragged out.

Lenses

Pad++ lenses can change the way objects look on the Pad++ surface [2][5][31]. In MSML, they can be particularly useful. For example, the data used to create the map of New Mexico consists of about 50K bytes of vertex data just to define the county boundaries and the city locations. There are many different kinds of data that one could be interested in displaying in relationship to a map of New Mexico (e.g., demographics of the local populations, geographical features of interest, etc.) If one were interested in not only providing an accurate street map to the Computer Science Department but also, say, an accurate count of the 1995 grasshopper population across the state, it would be pointless to include the geographical vertices twice. Instead, the same map can be used, and lenses supplied such that when viewed through the appropriate lens only the information of interest is visible. Figure 8 demonstrates the use of such a lens.

Annotations and Authoring

Because MSML allows users to interact with all elements of a WWW page, it provides a unique opportunity to explore annotating and authoring of WWW pages. Based on experiences working with children, we have come to appreciate the need for simple and intuitive ways to author WWW pages. One very natural authoring mechanism is to directly use and modify existing components of others peoples' web pages. Every web page then becomes a potential supplier of components.

We have begun to implement a drag-and-drop interface for authoring WWW documents. Figure 9 shows a page containing several Pad++ drawings. Just under that page, a composite figure was created by dragging elements out of the top web page (a kit of components constructed for young users to author zoomable hypertext stories) and putting them together. Our approach is to allow creation of web pages using the same direct manipulation techniques. In addition, we expect to provide layout support to facilitate creation of aesthetically effective web pages.

A direct extension of this authoring technique allows users to maintain local copies of WWW pages with added annotations. A goal, much in keeping with the kind of personal information environment Vannevar Bush envisioned, is to enable users to create, save, and share annotated databases of WWW pages. This is similar to graphical hotlists but with the important difference that with MSML, users can maintain not only the HTML data for a particular page, but also associate arbitrary Pad++-based annotations. Pad++ supports a variety of annotations: not only can text be added, but entire WWW pages can be added to other WWW pages creating hierarchical meta-pages; graphics can be added, for example, to indicate an interesting section of a particular page, even when that page is scaled very small. Our goal is to provide rich annotation facilities by combining Pad++'s dynamic multiscale annotation ability with a control system that supports creation of virtual documents in which documents and annotations are separately controlled and maintained. See [29] for an example of one such control mechanism.

To support effective views of collected web pages it is important to be able to show modifications over time. If one cares about a particular set of pages enough to include them in a personal collection it is likely that notifications of modifications will also be of value. We are exploring mechanisms to highlight changes. One technique highlights, by changing color or bounding with a rectangle, sections changed since some specific date (for example, the last time you viewed the document). Another uses scaling to show the history of changes. Older versions are shown at increasingly smaller scale. Marking changes is similar in spirit to efforts conducted by [35] using the WebWatch program. Unlike that effort we conjecture that one might not merely want to know that a document has been altered but also the details of the changes.

4. USER TESTING RESULTS

In November of 1995, we completed a pilot test of our zooming web browser. There were 14 test participants from the University of New Mexico community, equally split between the College of Education and the Computer Science Department. A majority used email regularly, but almost no one used the WWW as frequently. Over a third of the participants from the College of Education had never used the WWW. All of the Computer Science participants had used the WWW. Before the pilot test began, a 5-minute demonstration of our browser was shown to participants. Participants were then asked to use the browser for 30-60 minutes and subsequently completed a survey about their initial impressions.

We found users to be overwhelmingly positive about using the zooming web browser. We received such comments as:

> *"I think it's different than any other browser, and a lot more interesting to use."*
>
> *"It will take some getting used to, but to have the ability to create a tree of where you were is a great advantage."*
>
> *"It's easier and more friendly than Netscape. It is a little slow and jerky too."*
>
> *"I found it a very useful browser, and liked the hierarchical tree structures that are created. There is no need to click 'back' or 'forward' like you do for Mosaic or Netscape."*
>
> *"Easy access to previously viewed pages-- excellent way to view pages larger than the screen."*

The survey also asked users to select one or more entries from a list of possible short descriptions of their experience. The results are given in Table 1:

SHORT DESCRIPTION OF ZOOMING WEB BROWSER	# OF PARTICIPANTS WHO SELECTED IT (out of 14)
Interesting	12
Would try using it again	11
Enjoyable	7
Useful	7
Exceptional	4
Frustrating	3
Confusing	2
Would never use it again	0

Table 1: User Study results

When test participants were asked to describe what they most liked about using the zooming web browser, over 70% of the participants said they liked seeing the tree of where they were in the WWW and navigating by zooming. When participants were asked to describe what they liked the least, they commonly mentioned the speed of interaction. Users would like to have faster browsing tools, as well as the ability to delete a page from the tree structure. Based on this feedback, we have since modified the browser. In the newest release, WWW pages are displayed several times faster than during the pilot test and users now have the option of deleting any viewable page on the tree.

In the pilot test survey, participants were also asked how they thought the zooming web browser compared with other WWW browsers. Of the 14 test participants, a little over half felt qualified to respond. (A handful of the College of Education participants had never used another browser and therefore could not make a comparison.) The participants who responded were overwhelmingly more positive about the zooming web browser than Netscape or Mosaic. They felt that the browser had a better visual layout and was generally easier to use than other browsers. Surprisingly, only one participant felt that Netscape was easier and less buggy.

In summary, the results of our pilot test offer positive support for and constructive feedback about the use of zooming in a web browser. This feedback continues to inform our browser development efforts. We expect to continue testing with a more diverse population of users to better understand the problems and advantages issues associated with zooming.

5. A USER SCENARIO FOR THE FUTURE

With the technologies we are currently creating, we can foresee a time in the future when the following scenario will come to pass:

> It is morning. David Brooks enters his classroom. In an hour his fifth grade students will join him, but until then, David sits down at his computer, coffee in hand, to scan his favorite web pages. David begins by wandering the local museums' home pages. He knows that today he and his students will begin a thematic unit on dinosaurs. Before they arrive, he quickly drags various dinosaur images and text from different web pages, and creates a new student page. He decides that his page looks more like a wall of graffiti than a planned document. He wishes he had more time to animate the dinosaurs, design information lenses, and establish zooming links to other home pages. Suddenly it dawns on him, those would be great things for his students to do! He breaths a sigh of relief, sips his coffee, and waits for his students to arrive.
>
> Once all 21 students have been welcomed, David explains that thanks to their insistence, they will now turn their energies to learning about dinosaurs. The students clap and cheer. Once they settle down, David splits them up into design teams

of 3, and asks them to move to their computers. On each student team's screen is an image of David's dinosaur page. David explains how bad this page is, and asks his students to help him redesign it. He asks them to create animations, lenses, and links to other pages. The students eagerly work on their projects.

A week later, the student teams present their work to the class. The first group to go presents a page with four simply drawn dinosaurs. When a student zooms into the web page, the dinosaurs begin to move about. One of the students points out that they found information on the web that described how these dinosaurs moved, so they designed their beasts with this in mind. The team zooms in on one dinosaur, a tyrannosaurus rex. As they zoom in, the picture of the dinosaur disappears and text information appears. The student team explains that by selecting the highlighted word in the text it will take you back to the original WWW page that the text came from. As they explain, the text zooms out, a new tree link is formed, and a new page is zoomed in on the screen.

After much applause the next student team presents their work. They zoom in on their WWW page to display one large dinosaur. They explain that their project gives you lots of information on just one dinosaur. They begin by dragging various lenses from the side of their page. Each lens is shaped like the information it displays. The large tree-shaped lens when dragged over the dinosaur shows the types of vegetation the dinosaur eats. The sun-shaped lens shows the kinds of climates this dinosaur likes to live in. The team explains that by zooming in with any of the lenses it will take you to a WWW page with more information. They demonstrate this and start to uncover a tree of information.

David is impressed with his class's work. As each team presents, they offer more creative solutions than he thought possible. At the end of class he decides to ask the school principal if he can publish his students' research projects on the WWW. Not only does she agree, but she explains that the local museum has been looking for WWW pages created by students. She points out that this would be just the thing for their WWW section entitled: "A Kid's Tree of Knowledge" that is being designed to commemorate the fiftieth anniversary of Vannevar Bush's *As We May Think* article.

6. CONCLUSION

The World-Wide Web has become an important and widely used resource. Because of this, it is crucially important to address its usability. We have shown one promising technique based on zooming to better support web navigation. This technique was used to implement a prototype web browser that was found to be appealing to users. We illustrated extensions to the WWW authoring language to enable creation of more dynamic and interactive multiscale documents. Finally, we demonstrated how a direct manipulation authoring environment allows users to construct and modify web pages using items from existing pages.

Pad++ and the zooming Web browser will be made generally available in the near future. To find current information, send mail to pad-info@cs.unm.edu, or look at <URL: http://www.cs.unm.edu/pad++>.

7. ACKNOWLEDGEMNTS

We acknowledge generous support from ARPA's Human-Computer Interaction Initiative (Contract #N66001-94-C-6039). The test subjects from UNM's College of Education and Computer Science Department gave us valuable feedback. We appreciate the work of our collaborators at the New York University Media Research Lab, especially that of Jon Meyer, in helping us develop Pad++.

8. REFERENCES

[1] M.D. Apperley, I. Tzavaras, and R. Spence. "A Bifocal Display Technique for Data Presentation", *Proceedings of Eurographics '82*, 27-43.

[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, James D. Hollan, Ken Perlin, Jon Meyer, David Bacon, and George Furnas. "Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics", *Journal of Visual Languages and Computing* (in press).

[4] Tim Berners-Lee, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielsen, and Arthur Secret. “The World-Wide Web”, *Communications of the ACM*, August 1994, 37 (8), 76-82.

[5] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. “Toolglass and Magic Lenses: The See-Through Interface”, *Proceedings of ACM SIGGRAPH Conference (Sigraph’93)*, 73-80.

[6] “Blackbird”, <URL: http://www.microsoft.com/developer/intdev/bbdsht.html>.

[7] Nathaniel S. Borenstein. “EMail With A Mind of Its Own: The Safe-Tcl Language for Enabled Mail”, *ULPAA ‘94, Barcelona, 1994*. <URL: http://minsky.med.virginia.edu/sdm7g/Projects/Python/safe-tcl/ulpaa-94.txt>

[8] Vannevar Bush. “As We May Think”, *The Atlantic Monthly*, July 1945, 101-108.

[9] 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.

[10] Bay-Wei Chang and David Ungar. “Animation: From Cartoons to the User Interface”, *Proceedings of ACM Symposium on User Interface Software and technology (UIST’93)*, 45-55.

[11] Peter Doemel, “WebMap - A Graphical Hypertext Navigation Tool”, *2nd International Conference on the World-Wide Web*, Chicago, IL, 1994, 785-789.

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

[13] George W. Furnas. “Generalized Fisheye Views”, *Proceedings of 1986 ACM SIGCHI Conference*, 16-23.

[14] George W. Furnas and Jeff Zacks. “Multitrees: Enriching and Reusing Hierarchical Structure”, *Proceedings of ACM SIGCHI’94*, 330-336.

[15] George W. Furnas and Benjamin B. Bederson. “Space-Scale Diagrams: Understanding Multiscale Interfaces”, *Proceedings of ACM SIGCHI’95*, 234-241.

[16] George W. Furnas. “Effectively View-Navigable Structures”, Presented at *HCIC ‘95 Workshop*, Snow Mountain Ranch, Colorado: Human Computer Interaction Consortium Workshop, February, 1995.

[17] “Grow - The GNU Remote Operations Web”, <URL: http://www.cygnus.com/tiemann/grow>.

[18] “Java: Programming for the Internet”, <URL: http://www.javasoft.com/>.

[19] Wendy A. Kellogg and John T. Richards. “The Human Factors of Information on the Internet”, *in Advances in Human Computer Interaction, Volume 5*, ed. J. Nielsen, Ablex Press, 1-36.

[20] John Lamping, Ramana Rao, and Peter Pirolli. “A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies”, *Proceedings of CHI’95 Human Factors in Computing Systems, ACM press*, 401-408.

[21] Henry Lieberman. “Powers of Ten Thousand: Navigating in Large Information Spaces”, *Proceedings of the ACM User Interface and Software Technology conference (UIST’94)*, (short paper), 15-16.

[22] Jock D. Mackinlay, George G. Robertson, and Stu K. Card. “The Perspective Wall: Detail and Context Smoothly Integrated”, *Proceedings of CHI’91 Human Factors in Computing Systems, ACM press*, 173-179.

[23] Jock D. Mackinlay and George G. Robertson. “The Document Lens”, *Proceedings of the ACM User Interface and Software Technology conference (UIST’93)*, 101-108.

[24] Sougata Mukherjea, James D. Foley, and Scott Hudson. “Visualizing Complex Hypermedia Networks through Multiple Hierarchical Views”, *Proceedings of CHI’95 Human Factors in Computing Systems, ACM press*, 331-337.

[25] K. A. Oostendorp, W.F. Punch, and R.W. Wiggings, “A Tool for Individualizing the Web”, *2nd International Conference on the World-Wide Web*, Chicago, IL, 1994, 49-57.

[26] John K. Ousterhout. *Tcl and the Tk Toolkit*, Addison-Wesley, 1994.

[27] “Pad++”, <URL: http://www.cs.unm.edu/pad++>.

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

[29] Martin Roscheisen, Christian Mogensen, and Terry Winograd. “Beyond Browsing: Shared Comments, SOAPS, Trails, and On-line Communities”, Unpublished paper, Computer Science Department, Stanford University, Stanford, CA, USA.

[30] Monojit Sarkar and Marc H. Brown. “Graphical Fisheye Views”, *Communications of the ACM*, 37 (12), December, 1994.

[31] Maureen C. Stone, Ken Fishkin, and Eric A. Bier. “The Movable Filter as a User Interface Tool”, *Proceedings of ACM SIGCHI’94*.

[32] David Ungar and Randy B. Smith. “Self: The Power of Simplicity”, *Proceedings of OOPSLA* ‘87, 227-241.

[33] “The Tcl/Tk Project at Sun Microsystems Laboratories”, <URL: http://www.sunlabs.com/research/tcl/>.

[34] “VRML - Virtual Reality Modeling Language”, <URL: http://www.vrml.org/>.

[35] Kent Wittenburg, Duco Das, Will Hill, and Larry Stead. “Group Asynchronous Browsing on the World Wide Web”, *Proceedings of the 4th International World Wide Web Conference*, Boston, MA, International WWW Conference.

[36] “Yahoo”, <URL: http://www.yahoo.com/>.