

# HYPertext '87

## KEYNOTE ADDRESS

ANDRIES VAN DAM

I'm a Johnny-come-lately to hypertext: I didn't get started until 1967, and what is especially fun about being here is that I can pay public tribute to the two real trailblazers who have inspired me and hordes of my students who have gone off to do their own independent hypertext projects. The first is the incomparable, one and only Doug Engelbart, who has been working at this since the late 1950s. Many people don't know that. Some of them may go back in ancient history and remember his mind-blowing demonstration at the 1968 Fall Joint Computer Conference, but at that point he had already been working in this area for a decade. And he invented just about everything the rest of us have been doing since then. I will just mention two of his major contributions.

The first is office automation; he was doing office automation, in particular word processing, before the terms had even been coined. IBM invented the phrase, in connection with the MCST, the magnetic card selective typewriter. Word processing was the right term, since words were *all* you could process. However, you were lucky if you could replace a small word with a larger word and not have it flow off the edges of the card. At that time Doug Engelbart was really working on idea processing. He was, of course, the inventor of outline processing, as it is known today. He had links, he had text searches with a variety of wild-card options. How many of today's commercial hypertext sys-

tems let you perform text searches with wild cards and all kinds of other conditions? Not very many, I believe. He spent a great deal of time on the user interface. He had a variety of interesting command language shells that were at once the singular strength of his NLS system and, to those who were not willing to invest the time to learn them, also a weakness. He invented the mouse and the five-key chord input device and lots of other things.

So there is this whole catalog of stuff that Doug's group at SRI implemented in the middle to late 1960s on a little timesharing system on the SDS 940, the kind of CPU that today would fit in your wristwatch. All that functionality ran on a computer of that power!

The second major contribution, and the one that most people don't know at all, even if they know something about Doug's efforts toward the augmentation of human intellect, is that he is the father of software engineering in the modern sense. Long before scientists such as Dijkstra and Bauer started writing about formal software engineering, Doug and his crew had been living it. This really was a bootstrap community of tool builders and tool users. So on that little timesharing system on the 940, they had meta-assemblers and compiler-compilers and ways of generating special-purpose problem-oriented languages. They had any number of tools and understood the value of building tools. To me that was a revelation. In my group at Brown University we were assembly-language hackers, and what was important to us was efficiency and writing the tightest possible code so that we could get sophisticated programs to run in real time. The whole idea of spending a lot of time and energy on first building tools was really novel.

---

Photos courtesy of Steven Feiner, assistant professor, Dept. of Computer Science, Columbia University, New York, N.Y.

---

© 1988 ACM 0001-0782/88/0700-0887 \$1.50

So much for my paean of praise to Doug Engelbart. I think we are all here because of him and also because of Ted Nelson, the second trailblazer, who coined the word “hypertext” and dozens of other words—being wordsmith and master showman par excellence and also a polemicist of the first rank. Ted is a self-proclaimed visionary who deserves the title, and he turned on generations of people with *Computer Lib/Dream Machines*, a landmark work that still today—I reread a lot of it just a few days ago to prepare myself for this talk—is good reading. Ted coined that wonderful phrase, “If computers are the wave of the future, displays are the surfboards.” Well, I’ve used that bon mot ever since, and I think he is absolutely right: displays *are* the way to go. But one of the neat things is that we are discovering lots of other media that fit in: it’s not just display technology, it’s not just text and graphics any more. Video, high-quality sound and other media are also available now.

Another thing we should thank Ted for is that he did not just say, “branch, link, make arbitrary associations.” He tried very early to impose some discipline on linking, and introduced us to such wonderful artifices as stretch text, text that elastically expands and contracts in place. In other words, you don’t select something and then it blows up to an alternate statement or adds a level of indentation; no, the text should expand and contract smoothly, and you might use a lever to make things get more or less terse. I have not yet seen an example of this, and it’s really tough to write for, but it’s a very interesting idea.

Ted also talked about performing hypergrams: a picture annotated with text whose components you can point to and they will perform—animate themselves, for example. This was a decade before the MIT Media Lab did this sort of thing. Also I really think Ted deserves the credit for thinking multimedia so early. He also talked about zippered lists, collateral text and all kinds of other weird and wacky ideas, some of which are workable, perhaps, some of which are not, but all of them are sure stimulating!

One of the most important things he taught me was that this is a new medium and you really can’t be constrained to thinking about it in the old ways. Don’t copy old bad habits; think about new organizations, new ways of doing things, and take advantage of this new medium.

The other important thing he really led me to focus on is that there is a great application for hypertext and hypermedia, not just for research, not just for maintaining your own personal database, but in teaching. And that is one of my major themes here: how we have used hypertext and hypermedia in teaching at Brown University.

Well, here we are at the first hypertext workshop and we have to ask, perhaps rhetorically, has hypertext arrived? has the millennium arrived? And of course there are several answers to those questions. The state of the art is reviewed in Jeff Conklin’s excellent article in *IEEE Computer* of September 1987. There are also

articles in the press hyping hypertext, and about the hype in hypertext. Conversely, some of the recent major books on computing in the humanities don’t mention hypertext, don’t mention hypermedia, don’t really deal with the issue of branching texts. They are more concerned with typography, telecommunications and other issues that are very important. Even this 1986 book by Eugene Provenzo, *Beyond the Gutenberg Galaxy*, which is delightful reading, does not mention it. And the forthcoming book by Christopher Turk about humanities computing, *The Computer and the Scholar*, does not devote a chapter to this phenomenon, although electronic communication is certainly mentioned a lot, as are tools for processing text.

---

*One of the most important things (Nelson) taught me was that this is a new medium and you really can’t be constrained to thinking about it in the old ways. Don’t copy old, bad habits; think about new organizations, new ways of doing things, and take advantage of this new medium.*

---

When you look at the publicity and at systems such as Xerox’s NoteCards, Owl’s Guide, and especially Apple’s HyperCard, you get the impression that we are about to go over the knee of the exponential curve. I really believe we are. HyperCard in particular, despite all its limitations, is beautifully engineered, and has a wonderful user interface, especially for hypertext-style linking. It will really acculturate our computer user community. It is simple enough, despite its complexity, that a lot of people can get access to it at a relatively simple level. There is a lot of traffic on networks already about it and people are exchanging stackware. I think it will be a mass-media cult phenomenon, rather like MacPaint and MacDraw. I think Apple has done a really brilliant job of getting it out there, and although it’s a fraction of what Doug and Ted and others of us believe to be the potential of hypertext or hypermedia, I think it’s very strong and will do a lot to get people interested in our field. Presumably, it will be improved in future generations.

The success of this workshop also says we are about to go over the knee of the exponential curve. There is a staggering amount of information in the participant statements and in the papers, and many good and interesting papers. I learned a lot from just browsing through them and reading a few carefully. Frank Halasz’s paper is especially recommended, in terms of stating problem areas and an agenda for future problems we ought to be tackling (See “Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems” on page 836.)

So, while hypertext is not in the humanities comput-

ing books yet, which means that humanities scholars by and large do not yet know it exists, it will come through the HyperCard phenomenon, through the proceedings of this workshop, and the networking that this workshop will engender. So my summary is that no, the millennium has not arrived yet, but we are about to go over the knee.

Next, let's ask what is different about hypertext? Is there really anything new to it? You can trivialize it and say, "Nothing new, we've had network databases all our lives, what's the big deal?" Or you can hype it and say that it is an intellectual breakthrough. Or you can simply say there are lots of interesting ideas, but there is also a lot of hard work to do to make it really happen on a broad scale. Hypertext is basically clay, and we have to mold it; that is what this workshop is all about: starting to mold that clay.

Why did it take so long to have this workshop, to have HyperCard, when the technology certainly has been out there and there have been a lot of proof-of-concept demonstrations? Well, the first reason is the classical inertia problem. Why did it take twenty years for Doug Engelbart's mouse to be commercialized? One reason is there is tremendous inertia in this so-called progressive field. New ideas take forever to be popularized. The second reason is, of course, that there are technology problems. It takes a long time to develop something as cheap and as user-friendly as the Macintosh, for example, and to displace the idiosyncratic interfaces we have all created and been forced to live with. Now the technology is definitely here, and there is certainly no excuse for waiting any longer.

Next I'm going to give you my own personal view of some hypertext chronology at Brown University and then do some wrapping up. I ran into Ted Nelson completely by accident at the 1967 Spring Joint Computer Conference, and gossiped with him about what we had both been doing since we left Swarthmore College. He told me about his ideas on hypertext, and one thing led to another and Ted started coming to Providence, using, as he is proud to say in *Computer Lib*, his own money. We started working on the Hypertext Editing System, which was essentially dual-purpose. One purpose was to produce printed documents nicely and efficiently, since at that time the technology on IBM/360 systems was batch cards for editing (mag card selectrics were not yet common). But the main purpose was to explore this hypertext concept.

I want to mention a couple of numbers, just so that you can size the system. We ran our 2250 graphics display application in a 128K partition of a multiprogrammed operating system, on an IBM/360 Model 50 with 512K of memory, our mainframe at that time. An IBM/360 Model 50 is slower than a vanilla Mac and has less memory. Yet in that one partition we ran the Hypertext Editing System, and there was a complete timesharing system in another partition. There was no virtual memory, everything was done with software paging.

The undergraduates programming the Hypertext Ed-

iting System as a bootleg graphics project were paid by my IBM graphics contract. When our project monitor, Sam Matsa, saw it, he liked it, so we came out of the closet and started showing it around at a variety of sites where IBM had large customers. It was also ported to a number of university sites. Even after the project was frozen and we went on to the next-generation system, it was sold by IBM (unbeknownst to me and Ted and others who had worked on it) to the Apollo mission team at the Houston Manned Spacecraft Center and used to produce documentation that went up with Apollo, I'm proud to say.

Here are some technical features of that early system. It had arbitrary-length strings rather than fixed-length lines or statements, and edits with arbitrary-length scope, for example for insert, delete, move and copy. It had unidirectional branches automatically arranged in menus. It had splices that were branches invisible to on-line users that allowed the printer to go through a branching text. It had text instances. Some of the papers here discuss the differences between inclusion and reference. Instances are references, so that if you changed, for example, a piece of legal boilerplate that was referenced in multiple places, the change would show up in all the places that referenced it. Instances are a standard idea from computer graphics—no big deal.

Our data structure was a pointer-rich data structure on arbitrary-length pages. Edits were done by pointer manipulation, not, in general, by character manipulation. This technique was, in effect, a precursor of the piece-table technique, where you work with pointers to text rather than the raw text itself. We did software paging of arbitrary-length pages in this 128K partition, and that turned out to be too complex. Page-replacement algorithms when the slots have variable size are just not very manageable. We used Text/360, which was a batch formatter rather like MIT's Runoff, to produce hard copy. And so we could fulfill both of our objectives, producing hard-copy documents and fooling around with hypertext.

Let me tell you a little bit about the fooling around with hypertext. Ted's schematic of his hypertext of patents had a kind of hierarchical structure and then cross-references. These implemented the cross-references in the patents, and we also had various forms of indices; the real diagram was about the size of a large blueprint, as I remember, and hard to navigate on paper and on-line. We had already come to the point where Ted, who designed it, was able to go through the hypertext pretty well, but some of the rest of us had difficulty following it—it was not exactly obvious where you were. This, of course, is the classical lost-in-hyperspace problem, which has been mentioned by one and all; I won't elaborate on it here because it is amply discussed in the *Proceedings*. We already started getting the notion that the richer the hypertext, the greater the navigational problem. But we arranged careful demos in which we knew exactly where we had to go, and people were impressed.

In early 1968 we made the rounds of a number of large customers for IBM equipment, for example, *The New York Times* and *Time/Life*. And we found that our system was essentially too complex for them to understand. Remember that these people were producing magazines and newspapers and other forms of printed material. At most they had typographic programs that set type and maybe some software that did display ad management. But the idea of sitting on-line behind a tube and actually authoring and editing and rearranging and cross-referencing really was more than they were willing to believe you needed to do or should do. It was “very interesting.” It was futury and researchy, and I remember this particular demo we did at *Time/Life* when our audience said, “That’s great, but it will take us at least 10 years before people will be willing to sit down behind tubes and do anything on-line.”

---

*In 1968, I finally met Doug Engelbart and experienced his landmark demonstration. He was doing multiperson collaborations using all of his tools on-line interactively—it was a tour de force.*

---



Fortunately, they were wrong, and by the mid 1970s such places as *Newsday* on Long Island had bought precursors of the ATEX text-editing systems, and, much sooner than anyone would have predicted, this very conservative field of newspaper and magazine publishing switched over to editing on-line. All those people who sat down and typed with two fingers did in fact learn to organize their thoughts on-line on a VDT, despite dire predictions that they never would.

The other thing I learned was something about the art of giving demos: use progressive disclosure, don’t show it to them all at once. We used a standard 32-key IBM function keyboard, used in every computer-aided design installation in the world, and all the engineers were perfectly used to it, loved it, had no problems with it. But when we gave demos to people whose business was words, not engineering designs, they would freak out—they couldn’t handle all those editing buttons, one of which brought in another overlay of 32 formatting functions. So I learned very early to make a plastic overlay which I essentially used to cover up all but five of the editing buttons: insert, delete, move, copy and jump. Then we would do an entire demo for half an hour or so with that, let people play, and then we would say, “But wait, there’s more . . .”. And then we would play peek-a-boo, strip off the first overlay and, lo and behold, there was another row of function keys. And so by doing progressive disclosure with a sequence of increasingly revealing overlays we managed not to swamp and panic people.

The last lesson learned during that period: real es-

tate, screen real estate. We saw that people would sit behind a 2250 Mod 4 display with software character generation, where they could put up teeny tiny 8- or 10-point text with the screen flickering wildly—three, four frames a second, sometimes two. And they would prefer that to a smaller screen or larger, steadier characters, simply because context is so extremely important. It drives me wild to sit behind a small-screen Mac and be restricted to this little 3” × 5” card image, not even able to scroll around on a drawing to get more of a view. We have to deal with this problem of getting more screen real estate so that it feels like something we normally work with, even something as limited as 8½” × 11”. We have to get bigger displays, and we must have easier ways of moving around on the displays we do have, because these little tiny windows just drive you crazy. I’m going to come back to that.

In 1968, I finally met Doug Engelbart and experienced his landmark demonstration at the Fall Joint Computer Conference, about as gutsy and risky a demo as I have ever seen. He was doing multiperson collaboration, using all of his tools on-line interactively—it was a *tour de force*. He did outline processing, browsing and jumping around through multiple files with text and graphics, using key words and other view specs to act as viewing filters over his data, using text-searching capabilities, and so on.

Later that year I went on with my students to design FRESS, a File Retrieval and Editing System. My design goal was to steal or improve on the best ideas from Doug’s NLS and put in some things we really liked from the Hypertext Editing System—a more freeform editing style, no limits to statement size, for example. But the intellectual debt is clearly to Doug’s NLS.

One of the things I worked very hard on was output-input device independence and the idea of a logical device. We wanted to get this system to run on anything from teletypewriters, which were very common at that time, up to and including minicomputers with multiple windows on vector displays. We had a 16Kbyte IMLAC mini with a vector display that was used very extensively—a single one, because that was all we could afford.

So we developed the notion of virtual output and input devices, along the lines of Fortran logical I/O units. Virtual input devices were an important part of a graphics subroutine package called GPGS that we designed in 1971 in the Netherlands as part of a three-university consortium. And that idea went into the CORE, GKS, and PHIGS standard graphics packages and now into PHIGS+, the latest graphics package specified by an ad hoc standards group in 1987, almost two decades later.

FRESS was of course multiterminal, it supported outline processing, arbitrary-length strings, edits, etc.; it had no size limitations. I really believe in that—I think it is important not to feel yourself constrained. One of the really important ways in which unconstrained size and scope ought to work is that it should not interfere with performance. In FRESS, to a first approximation,

you could not tell the difference between working with a two-page printed file and a 200-page printed file because of the software paging scheme we used. And yes, of course, some tables grow with file size, but in general the system was just as peppy with large files as with small files, and I think that is really important for users. While keeping a lot of small files may be a feature for some people, it's a bug for many others, and the system should not force you into an unnatural usage pattern.

We also went from unidirectional links in Hypertext Editing System to bidirectional links with explainers in FRESS. I think bidirectionality is important: the 'come from' is as important as the 'go to.' Key words were possible on every element, both for on-line and off-line trails. Links could be "typed" with these key words. We also experimented with protection down to the character level, and every single one of a hundred-odd functions could have a bit in a mask that said whether you could perform that function on this block of text or not. It was overkill, and we haven't done it since, but it certainly was interesting.

We had an ability to see the structure space, a visualization of all the structure in the text, the outline structure and the cross-reference structure. You could do structural rearrangements in that structure space in a quick overview mode and you would thereby induce those same edits in the text itself. And of course edits made in the text would also appear in the structure space. That duality was a very useful and popular feature.

The most popular feature, however, was undo. I will claim that, to the best of my knowledge, FRESS was the first system to have an undo. We saved every edit in a shadow version of the data structure, and that allowed us to do both an autosave and an undo. I think the most important feature in any system built today has to be indefinite undo and redo. One level is better than zero, but not enough, and wholly inadequate for serious work. It doesn't matter how it's done, whether it's journaling, transcribing done with inverses, whatever, but you've got to have it.

We had escapes to the command shell, so that you could do some useful work outside the system. That is another theme I want to return to: whether you should have a self-contained, self-sufficient hyperuniverse or should be able to go outside and do things you need to do that it does not (and should not) provide.

I want to tell you a little about the sociopolitical climate then, because these really are the barefoot-in-the-snow stories that most of you, fortunately, have never been exposed to. When we were doing this work at Brown University, nobody said "Hey, it's great you're building tools for humanists, that's wonderful, when can we have it?" In fact, quite the reverse. Those were the days of accounting on mainframes. You got a certain amount of 'funny money' allocated every year and you had to make it last, and if you ran out, tough. Maybe you could argue for a little more and maybe you couldn't. So it was extremely important to make our system efficient so that people could afford to use it on their funny-money budget. I had serious warfare with

the vice president in charge of computing about whether the software should even be allowed on the system, because if it were on the system, then people would use it. And that would subvert the true purpose of computers, which was to produce numbers for engineers and scientists. He said, literally, "If you want to screw around with text, use a typewriter." I essentially had to blackmail him by saying there would be a revolution by the humanists on campus if they found out how much the engineers and scientists were spending on computing when they couldn't have any.

These were the best of times, these were the worst of times. They were the best of times in that there were a lot of technology and "proof of concept" systems out there. They were the worst of times because we really had to fight to get these ideas recognized as legitimate fields of inquiry and to get real users. In about 1972 I watched Doug do his standard 90-minute pitch with video and live demos to a group of DARPA contractors who were the best and the brightest in the country at that time. And one of the very best people said, basically, "Doug, I don't see what you're doing there that I can't do on my glass teletype with a good line editor." But Doug, fortunately, kept going, we kept going, other people did. And by the mid 1970s we finally got around to doing what I had always wanted to do since Ted first tuned me in to the idea, which was try out this on-line stuff in education. So, very briefly, I'll describe two experiments. In one, funded by the Exxon Education Foundation, a physicist and I did a course called *Man, Energy and Environment*. Students did a lot of reading of hypertext on-line about the subject, but no writing. Then we did a much more ambitious experiment in the following two years, funded by the National Endowment for the Humanities. For this English poetry course we used a very large hypertext with well over a thousand links. Three times a week students had to sign up for an hour each on our one and only Imlac graphics workstation and do their reading and their commenting on-line, following trails, making trails. We used a kind of progressive disclosure: the first time through they saw the poem they were supposed to critique and analyze, with no references. The second time they saw it with a few links to other poems on the same subject or by the same poet. There would also be some word glosses, some professional analyses, but still not very much context. And they would be reviewing what other students had written on the first pass, and the teacher's and TAs' comments as well, and then they would form a new opinion of what they had read. And then they would do that a third time, when they had yet more access to what people had written communally and what had previously been put in the database. It was very interesting. People loved it, despite the fact the system went down a lot, that it was hard to get at it, that you had to schedule time. And this "communal text," as it was called by the poetry people who wrote about it later, became very rich in additional annotations. Electronic graffiti, as I thought of them.

The reason I encouraged such annotations was that I



remembered that when I was in college with Ted, I would always grab the dirtiest copy of a book in the library, rather than the cleanest one, because the dirtiest ones had the most marginalia, which I found very helpful. It really worked here: on average, students wrote three times as much for both analyses and informal discourse as they did in the control group, and that pleased the faculty who were very much concerned with encouraging oral and written expression. One of the things we found is that people express themselves differently. Some people who were very articulate in a classroom setting were pretty silent once they got behind the tube. And vice versa, so a lot of shy violets really became vocal once they got behind the tube.

What we discovered from that experiment is that people could follow trails and enjoyed it. But what we did not see was a lot of people blazing trails. There wasn't enough time, the interface wasn't good enough, response wasn't fast enough—a variety of reasons. So we never really proved a central hypothesis, that people can and will blaze trails, not just follow them. I will make a statement that says, despite the experience with Xerox's NoteCards system, despite whatever other experiences there are out there with programming environment browsers and so on, by and large the hypothesis remains unproven that, with little guidance, people can construct really good trails, really good webs that help them and help other readers. I think we still need to test that hypothesis in a major way.

The next thing we did after FRESS was put to bed in the late 1970s was to make a rather different hypertext system, our third by that time. What we were interested in then was power tools for producing primarily graphical documents. I want to describe to you very quickly some of the things we did in our electronic document project.

There were three components to the system. One was a viewer's component in which prepared screen pages could be viewed. Figure 1 shows four iconic buttons, each one a composite of individual images drawn by hand and then, with the help of the authoring component, put together on a screen page and buttonized. This is typical of what a maintenance and repair technician would see as a typical page in the little maintenance and repair Dynabook icon. The other two components were for the author. The first was for making graphics for individual pages or their parts, a MacDraw–MacPaint combination. The component most interesting for this audience, shown in Figure 2, was the hypertext creation component for constructing pages, chapters and links. It was multiwindow, so that a page under construction could occupy either the entire screen or a portion of it. This allowed multiple pages or even chapters to be visible simultaneously so that you could create links from one to another. The whole interface is manipulated directly: there's almost no typing and lots of pointing at objects and manipulating them. This was back in 1979 when that was not so common.

We spent a lot of time figuring out how to elide, that is, hide, information graphically. We had a 'detail but-



FIGURE 1. Document Presentation System Page

ton' that let us view things at varying levels of detail. So the author could move these windows around, look at pages and chapters at arbitrary levels of detail, iconically create various kinds of buttons and specify actions to take place when the reader invoked a button. Such actions could include animation and taking a link to another page. The model for this hypertext is more a finite-state automaton in which transitions take you from one mode to the other rather than of a static

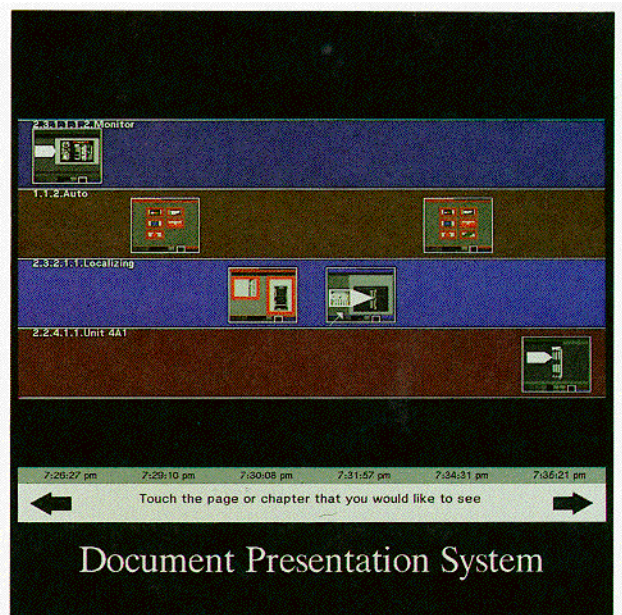


FIGURE 2. Document Layout System Interface

hypertext with static links. In particular, using a key-wording facility, you could cause different parts to appear on the same page as a function of the keywords encountered during traversal. So on two successive accesses a page might look quite different through hiding information or showing new buttons.

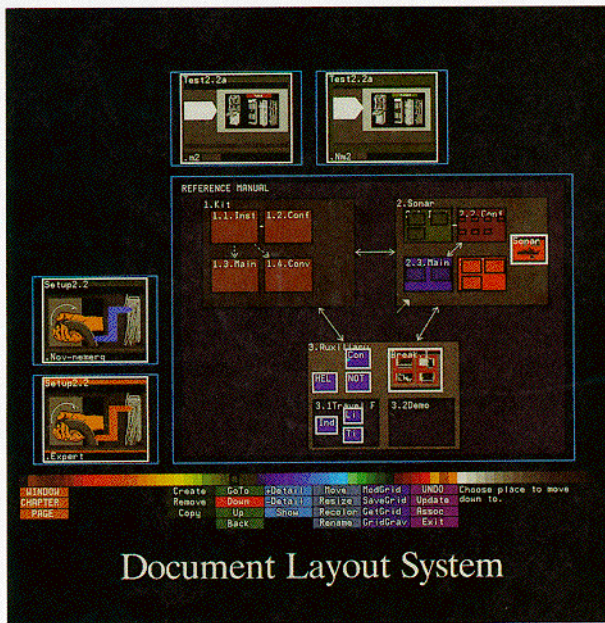


FIGURE 3. DPS Timeline

The system had three automatically created navigation aids. First, it had a time line, as shown in Figure 3, that you could use to travel back in time, consisting of buttons of miniature page icons stamped with time and date, with color coding keying you to chapters. This is a little bit like the 'recent page' in HyperCard, but here we are time traveling: we are not just going back to an image, we are going back to the state that accompanied our view of that image, because a page, as you remember, can change over time. Another was the 'neighbors' display: the current page is displayed in context with (on the left) a filmstrip of all the places you could have come from, and (on the right) all the places you could go to, and again you can pick any of these miniatures and go there directly. In general, this could be an arbitrarily complicated display with dozens of sources and destinations; then you use iconic scroll bars to move over these filmstrips. The third navigation aid was a visual index of buttons of page miniatures arranged by key words, color-coded by chapter.

We learned some new lessons from this third system. It had great power tools for the author, but it still involved a tremendous amount of hand work. What you really would like in making fairly regular documents, like maintenance and repair manuals, is to drive the creation of the pictorial or textual, audio, video, etc., explanations directly from deep knowledge about the problem domain, the particular problem to be solved,

the design rules for creating explanations in the various media, and the user of the manual. You want, in effect, to produce an automated authoring capability in which most of these things can be driven from these knowledge bases. Steve Feiner's Ph.D. dissertation and his work since then describes such automated authoring.

The next system I would like to mention briefly is not a hypertext system, but it formed our thinking about ingredients at the nodes of a hypertext. This is the Balsa (Brown ALgorithm Simulator and Animator) environment, created in 1983 by Bob Sedgewick, who is now chairman of computer science at Princeton University, and Marc Brown, who last year received an ACM distinguished dissertation award for this work. In the course of a lecture we use Balsa on a network of workstations in the classroom to look at dynamic visualizations of programs implementing algorithms and data structures. If a picture is worth a thousand words, a dynamic picture of time-varying objects is worth a thousand static ones. We need dynamics at the nodes, not just static pictures and text. Something one learns quickly about user-controlled real-time animation is that hardware power is really essential. If it takes 10 seconds to put up the next picture because that's how long the hardware needs, you do not have kinesthetic feedback, you do not have smoothness and visual continuity, it is not responsive. It makes a great demo, but it drives you crazy in real life.

We have been using electronically assisted teaching now for five years and our new building contains two workstation auditoriums. Balsa has been integrated into a variety of courses in computer science and math, as well as in neural science and even political science. These experiences led to Brown's wholesale commitment to workstations and in fact to the creation in 1983 of IRIS, our Institute for Research on Information and Scholarship. IRIS not only creates scholars' workstation software but also has a completely symmetrical program evaluation branch where social scientists with an interest in this area study needs, requirements and impact of this technology on scholarly work. And by the way, 'scholarly work' to us is not just the work of faculty members but also that of students—we're all scholars on this bus.

We set up IRIS at a time when there was a lot of enthusiasm in front-running universities for buying into the workstation revolution, looking at the Xerox PARC model of distributed computation and saying, "We want to deliver tools that really take advantage of the expressive power of such workstations." So IRIS and other organizations on other campuses were created to develop scholarly tools. I think, though, that there is still a paucity of such scholarly tools and we are at the very beginning of understanding what they might be.

I'm going to sum up what I think are some key areas we all ought to be looking at. Here are nine quick items. My first concern is that systems may develop great infrastructure, great hypertext glue, but what's at the nodes? As with the 2D animation in Balsa, I think



we should be aiming at user-controlled 3D animation, where the interactive user has control over what is being shown, the level of detail, the visualizations employed, and so on. This is just a wide-open area that needs a lot of study, not to mention vast hardware power.

Second, Ted talks a lot about the docuverse, a mythical entity out there that is all-inclusive and contains everything. But instead, right now we are building docu-islands; none of our systems talk to each other, they are wholly incompatible. So we are all working the same agenda, more or less, but we can't exchange stuff; there is no exchange format, there is no universality, and furthermore, our systems are closed systems. In a sense, they are making the same mistake as the all-in-one environments in personal computing. Yes, they give you a word processor and a spreadsheet editor and a business graphics package, etc., but none of them are really satisfactory. And our experience with FRESS, where we had to escape to command language, showed that it is really important to be able to go outside. So it's not enough to bundle the HyperCard package with every Mac you buy. It really ought to be migrated down, become part of the toolbox, so that application programmers can take their applications and take advantage of a standard linking protocol that works within and between applications.

---

*If a picture is worth a thousand words, a dynamic picture of time-varying objects is worth a thousand static ones. We need dynamics at the nodes, not just static pictures and text.*

---



So I'm going to raise a red-flag word: standards. I'm a firm believer in standards. And everybody will say it is absolutely premature to standardize when we don't even know what the hell we're talking about. We are still in the experimental phase. I believe that. But if we don't start thinking about standards, five years from now we are going to have a wealth of these little docu-islands which are totally incompatible, and that's crazy.

Point number three is size. We are still in the toy problem stage. There has not been a decent-sized hypertext built yet. And we won't know what it is like until we deal with the kinds of documentation problems that people in the real world deal with. People have graphed the number of pages of technical manuals for fighter aircraft against time. In World War II, we had 1K page documentation per fighter aircraft; in Korea, 10K; in Vietnam, 100K. It's an exponential curve. The F-16 has 600K pages, and the advanced tactical fighter will probably have somewhere around a million and a half pages of technical documentation. It is, in effect, a giant hypertext that should be linked, and done with change control.

And that is my fourth point. We have to do something about versioning and change control in the large. But we won't have a fighting chance to kill paper and become an on-line community until we tackle the kinds of problems that paper solves for us today. Paper is convenient, you can carry it around and versions are easily created and maintained. Hypertext is not as accessible as paper, and it certainly isn't prepared to deal with the size problems as paper does.

---

*Remember how crazy everybody went when they got the ability to print multiple fonts? We got 'fontitis.' Then people discovered color screens and we got 'coloritis,' without any rules, without any design discipline. Now we've got 'linkitis,' and people with no graphic design experience or talent are going to throw stuff together and it will look terrible.*

---

The fifth problem is navigation, lost in hyperspace. Some people say we need content and structure query, we need virtual structures, composites; others say we need pruning. All of this is true. I find the notion of dynamically constructing a hypertext very intriguing, but very difficult to do in general. My point of view is that in a sense hypertext gives us a **goto**, and a **goto**, as we all know, produces spaghetti. At most we have invented the **if-then-else-if**, with hierarchy. That's our one structured flow-of-control concept, where you have some sense of what you are looking at in your hyperdocument. Well, we need to discover what the equivalents of other constructs are. As Ted did at the start, we have to invent other document forms that somehow become standard so that people have pattern recognition and say, "Ah, yes, I know how that one works." So we need new forms, new flow of control kinds of constructs besides just unbridled **goto**-ness of directed graph structures.

The other navigation aid we need is a tell-me-what-you've-got, which probably entails AI. Instead of just syntax, we need some notion of semantics at the nodes. We need to be able to derive new knowledge from old, using inference engines. Just storing billions and billions of facts that are tied together is not going to do it for us; that will just drown us in associations.

The sixth point is that we need hypermedia designers. Remember how crazy everybody went when we got the ability to print multiple fonts? We got 'fontitis.' Then people discovered color screens and we got 'coloritis,' without any rules, without any design discipline. Now we've got 'linkitis,' and people with no graphic design experience or talent are going to throw stuff together and it will look terrible. You all know that one of the interesting things that happened after Gutenberg



was typography. For a while after Gutenberg, type designers were using manuscript letterforms. And they were using erasers and paintbrushes to doctor the stuff the typesetting machine had set so it would look more like an old-fashioned manuscript and people would be familiar with it. It took a long while for people to understand that this new medium demanded a new style, a new typography. So typography is an invention, and it is nearly as important as the printing press in the first place. We don't want to mimic the old manuscript form in our hypertexts.

Aldus Manutius discovered modern pages. And pages gave us a new way to cross-reference, to talk about content, to index, and that technology took a while, but it made information a great deal more accessible. So we need people who are concerned about layout and design and typography of hypermedia and can think about classification and indexing and how we put things together.

But we don't want to put things together in such a way that there is *one* point of view, because if we've learned one thing from interactive tools up to now is that multiview is the way people work. You can not have it just one way. We need an update to Larry Tessler's "Don't mode me in." Jim Foley and I recently came up with "Don't metaphor me in." Don't give me a little card image and say, "That's all you've got, because that's what I thought you should want for your virtual shoebox." There have got to be multiple modalities and the designers have to be able to deal with that. So that was issue number seven: don't metaphor me in, don't give me only one way of looking at things.

Point eight: accessibility, portability—I want a laptop Dynabook. And I also want wallscreen-size displays. I want my whiteboard-sized hypertext display, and not until it's that big do I think we can really start working with hypertext. Hoker performs the following experi-

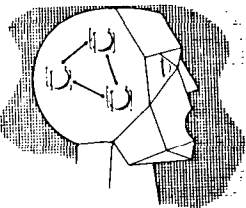
ment. Take *The Wall Street Journal* and move around a little cutout the size of a Mac screen and see how happy you are with *The Wall Street Journal*, versus looking at a page of it in its entirety and immediately picking up what you might be interested in reading. A very persuasive point.

And finally, point number nine: we must think about the sociopolitico-economic problems. We have not come to grips with issues of intellectual property rights and compensation. Congress and the U.S. Patent Office and copyright lawyers and so on do not understand what is involved here and need to be educated. If we shrink away from those issues and leave it to them, there is going to be absolute chaos. Ted, again, has written extensively on this issue and has postulated some interesting mechanisms for coping with the problems.

So there are essentially three classes of people. There are the visionary hypers who say, not only is the glass not half empty, it's overflowing with opportunities and possibilities and technology push of CD ROMs, 100 megabytes and 100 MIPS on your personal computer in just a few years, and so on. They are saying everything is great and we have a wonderful universe to explore.

Then there are the skeptics who say, not only is the glass not half full, but there was probably never any water in it to begin with. And I'm a third type sitting on the fence somewhere in between these two extremes, saying, I've used a lot of hypertext systems and I think they are neat and show a lot of potential. I think we are just at the very beginning and it's too early to rush to judgment, but it's clear that collectively we have a lot of hard but exciting work to do to make this technology work and to create what has been called the electronic Alexandria.

Author's Present Address: Andries van Dam, Brown University, Dept. of Computer Science, 115 Waterman St., Providence, RI 02912.

<b>COMPUTING TRENDS IN THE 1990'S</b>	<b>Conference Highlights:</b>	
<b>1989 ACM Computer Science Conference</b>	<ul style="list-style-type: none"> <li>• Quality Program Focused on Emerging Computing Trends</li> <li>• Exhibitor Presentations</li> <li>• CSC Employment Register</li> <li>• National Scholastic Programming Contest</li> <li>• History of Computing Presentations/ Exhibits</li> <li>• Theme Day Tutorials</li> <li>• National Computer Science Department Chair's Program</li> </ul>	
	<b>Attendance Information</b> ACM CSC89 11 West 42nd Street New York, NY 10036 (212) 869-7440 Meetings@ACMVM.Bitnet	<b>Exhibits Information</b> Barbara Corbett Robert T. Kenworthy Inc. 866 United Nations Plaza New York, NY 10017 (212) 752-0911
<b>February 21-23, 1989</b> <b>Commonwealth Convention Center</b> <b>Louisville, Kentucky</b>	<b>acm.</b>	