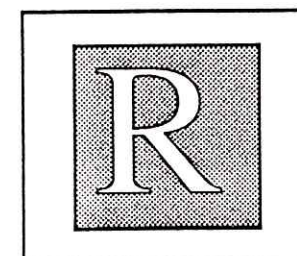
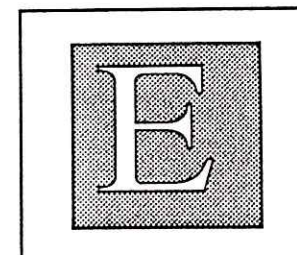
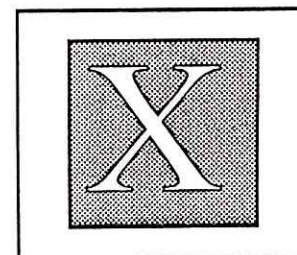
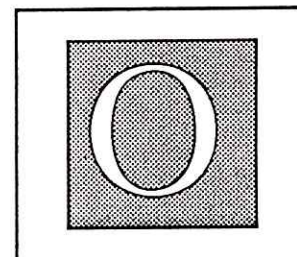
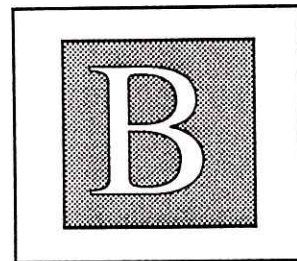


Social Niches for Future Software

Andrea A. diSessa
University of California, Berkeley

May, 1988



THE BOXER GROUP
Graduate School of Education
University of California, Berkeley
Berkeley, CA 94720
415-642-4206



This project was supported, in part,
by the
National Science Foundation
Opinions expressed are those of the authors
and not necessarily those of the Foundation

This research is supported primarily by a grant from the National Science Foundation to Andrea A. diSessa. We also gratefully acknowledge a contribution to support students from the Apple Corporation, and a contribution of equipment from Sun Microsystems. To appear in diSessa, A., Gardner, M., Greeno, J., Reif, F., Schoenfeld, A., and Stage, E., *Towards a scientific practice of science education*. Hillsdale, NJ: Lawrence Erlbaum.

Activity Structures and Social Niches

Considering that computers have been with us for some time, I am surprised at the primitive state of our critical capability to judge their usefulness in educational settings. What I see often are arguments at very general levels -- "computers are dynamic, interactive and can respond patiently in an individualized manner." To be sure, one can rely to some extent on such generalizations, but most confidently in retrospect rather than in anticipation of good educational outcomes. Engineering and science are driven by expectations and promises like these, but anyone who has engaged in design or science knows how these can fail in the details -- for some reason, the promises just don't get down to brass tacks; general goals and possibilities don't really tell us what to do. Critical judgments of particular software are also too often driven by face validity. It is as if one could see quickly and easily the features that lead to deep learning. One hears that the software contains a lot of information; it offers the user immense freedom; the simulation is dynamic and interactive; ideas are rendered concrete and manipulable; the program satisfies a checklist of "good educational software" rules and has a clever user interface.

I do not wish to dismiss any of these as unimportant. Yet at the same time, I wish to highlight a class of critical criteria that appears to be difficult or impossible to see "in the software." I can introduce the topic with a set of questions that I compulsively ask students when they present to me a piece of "educational software," say, a microworld, from which they are sure children will learn: What exactly is it that children will be doing with this software; what will *they* think they are doing with it; why should they do what you think they should do; and how is it, exactly, that they should learn from what they do? What happens if they have different purposes, and if they should see the program in a different light than you intend? Why shouldn't the child get the wrong answers just to see what happens, or turn your science microworld to artistic ends?

More compactly, I believe the single most important heuristic for evaluating software is to try to simulate the child's *activity structures* involving the full spread of his motivations, his capability to describe what he should be doing, what he can make of the things that do and don't happen in

response to his initiatives, and so on. In fact, I claim that these activity structures are the things to be evaluated, not the software *per se*, and that it is in these structures that we can see not only how well children will learn, but also what they will learn.

This is evidently a knowledge intensive evaluation technique. It must bring to bear everything we know about the consumers of our products, not only first impressions. This evaluation entails long-term trajectories and possibly subtle turns of meaning for people who may be very different from ourselves. If you cannot answer the above questions about how students will perceive and interact with software, you have not even begun to evaluate it. Quick arguments about face validity and first impressions are intriguing, but it is proper to remain skeptical until activity structures can be analyzed.

The title of this paper emphasizes that student/machine couples are not the only level of analysis important to this kind of assessment. Any personal activity interfaces with social levels of activity. Group commitments and patterns of cooperative learning always contribute another level of dynamic to activity structures. At the very least, classroom social structure both among children and between the child and teacher must allow what we judge to be productive individual activities. More profoundly, activities are deeply affected by social concerns even if they are engaged in individually. This becomes more evident if we lift our gaze from the sanctioned school and classroom culture to the family and peer group.¹ Motivations of individuals reflect communal judgments of what is appropriate in various circumstances, including whether it is appropriate to think or "try." Overall, we want to think about activity structures as fitting into niches defined by interest, need, capability and judged appropriateness, both at individual and social levels.

1. See Penny Eckert's discussion in diSessa *et al.* [to appear].

It is not my aim to contribute theoretical or empirical results to this important area. For that I must defer to experts such as Saxe and Lave.² Instead, I take a designer's point of view and through these concerns introduce the general frame that we (the Boxer Project Group at Berkeley) have been developing to anticipate and, in part, to evaluate the effects of the work we are doing in developing a general purpose computational environment. What follows are some of our particular images, our simulations of future activity structures involving the system we are designing. They may be less certain than *post hoc* studies. But as designers, we cannot escape the fact that the likelihood of our success depends, in large part, on our ability to imagine what has never been.

The Concept of A Computational Medium

The Boxer Project is aiming to test the feasibility of filling a rather grand and still hypothetical social niche, that for a computational medium. We are trying to produce a prototype of a system that extends with computational capabilities the role now played in our culture by written text. It should be a system that is used by very many people in all sorts of different ways, from the equivalent of notes in the margin, doodles and grocery lists -- all the way to novels and productions that show the special genius of the author, or the concerted effort of a large and well-endowed group. In a nutshell, we wish to change the common infrastructure of knowledge presentation, manipulation and development. More modestly, we want a general purpose system to serve the needs of students, teachers and curriculum developers, something that is so useful for such a broad range of activities that the community as a whole will judge it valuable enough to warrant the effort of learning a new and extended literacy.

Already the choice of target niche defines many of the properties of the system. Notably it will be complex and non-trivial to learn. Like written language, it will gain its foothold by being learnable in small chunks that are by themselves useful, and by paying back the substantial

2. See also diSessa *et al.* [to appear].

investment made to learn it over a lifelong usefulness. My most compact description of the learning process is learnability through "continuous incremental advantage;" the system at each stage gives the learner an additional capability she wishes to have by learning a little bit more about the system, throughout the whole course of intellectual development. The child gains by learning a word, say, "cookie," and the adult gets a job by filling out the application competently, or gains in professional prestige by eloquently expressing her ideas.

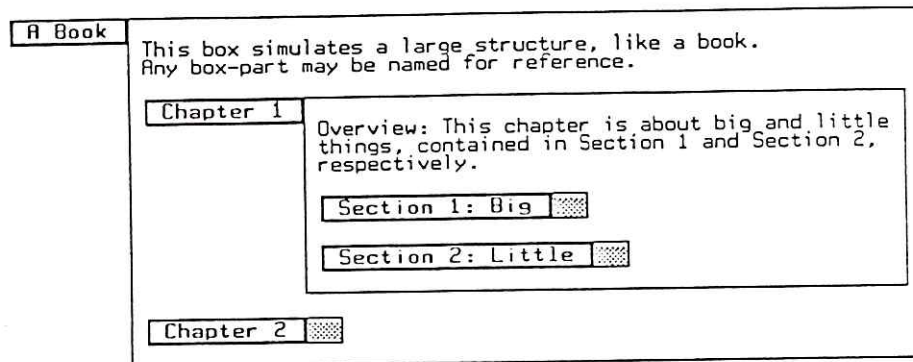
What does a computational medium look like? Begin with written text, for it is certainly true that a computational medium would be forsaking a fundamental and established niche if it did not build on this already well-established medium. With only modestly enhanced capabilities, including automated ease in editing and formatting, computers in the form of text processors are already taking over in professional settings, and even in schools, from pens and typewriters. This is, however, only the beginning of what we visualize as a computational medium.

The next jump comes when we free text from its purely linear form. Hypertext has captured the imaginations of media enthusiasts, and similarly we see a move in this direction as providing important new possibilities (so long as the extension is not so complex and specialized as to preclude use in the simple manner that a basic text processor represents). What hypertext supplies is the ability to *chunk* information into packages, and to *link*, providing non-linear connectivity from package to package so that users can peruse a hypertext in a broadly flexible way, and so that, on the other side, writers can collect and structure their ideas gradually, capturing in the physical layout an appropriate slice of the organization of those ideas, not just a linear slice.

For Boxer, our new system, we have added two simple hypertext capabilities to text. Text can be chunked into boxes, which are always part of text in some larger box. Thus, the basic structure of Boxer is a hierarchy of text and boxes (that contain more text and boxes) in arbitrary combination. Any box can be named, if that is deemed appropriate.

Boxer can be thought of initially as an extended text processor. In addition to ordinary text, one can insert at any point a box to "chunk" additional material.

A unit of thought of arbitrary size can be put in a box and moved or deleted as a unit.



Note that detail in the book has been suppressed by shrinking lower level boxes and boxes in which we are currently uninterested.

Finally, one can view and "zoom" to remote boxes through ports like the following. The port happens, in this case, to view a box that is also currently visible near the top of this figure.

port

A unit of thought of arbitrary size can be put in a box and moved or deleted as a unit.

Figure 1. Boxer documents are a simple form of hypertext, chunked and hierarchically organized with boxes, and cross-referenced with ports.

Figure 1 shows some text and boxes that were merely typed directly into the system. When the user wishes to have a box-chunk of text, she just presses a key that makes a box. The box expands as text is typed or moved into it to accommodate whatever the writer wishes. From one point of view, boxes are just "large characters" that can be selected, deleted, cut or pasted the way other text is in a word processor. The small grey boxes in the figure have been shrunk, hiding the detail contained inside. Shrinking and expanding are key-stroke (or mouse-click) operations. The user can effectively zoom into any visible box, temporarily forgetting its context, by expanding the box from its normal size, after which the frame of the box occupies the full screen. We believe Boxer's capability to suppress or show detail with nothing more than a key-stroke will be an important but easily understood expressive extension to ordinary written text. It can carry many important ideas -- notably, it is a much better representation than static,

linear text of levels of understanding and the process of zooming into details, yet being able to "shrink" back to get an overview when necessary.

Linking, that is, direct access to remote corners of a Boxer document is provided by a kind of box we call a port. Ports are simply views of some box, the "target," which may exist anywhere else in the user's Boxer universe. An editorial change to either the port or its target results in changes to both. Ports make good cross-referencing structures, and one may trivially transport oneself to the place where the target of the port exists if one wants to see more about that context. The box at the bottom of Figure 1 is a port whose target is the top-most box in that figure.

Of course, graphical images are important, and Boxer has graphics boxes that contain pictures. Graphics boxes also contain mobile and interactive entities we call sprites, which can draw and can be simply taught to respond to clicks from the mouse. These are the basis for Logo turtle activities, but also they add the capability to define modes of interaction with Boxer that were not built in. Thus we have extended text now to include images that can be both dynamic and interactive.

Finally, we add to what we have so far the capability to program. This already separates Boxer from essentially all hypertext systems. However, Boxer is unique in that not only does it contain a programming language, it *is* a programming language. Incremental learnability is provided in that anyone who can create and inspect a simple document in Boxer knows almost all the mechanics of creating and inspecting programs. Further, joining programming to (hyper-) text processing adds utility and flexibility. For example, anything that is typed into Boxer is accessible data, to be manipulated at will. So, if we keep a personal journal as a box containing entries (more boxes), one can write a simple program to rearrange all the entries in the box into some other ordering, say, based on topic headings rather than chronological order. Programming is the glue that permits arbitrary recombination of the capabilities of the system or any documents written in it, giving as full control to the user of a document as to its author.

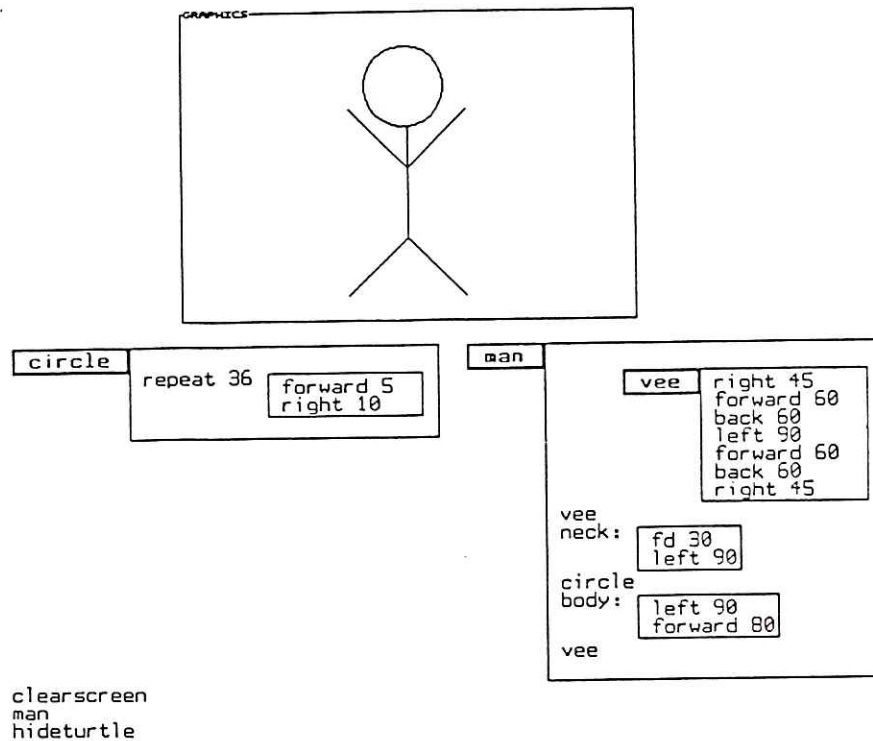


Figure 2. Program definitions appear precisely where they are useful. A general utility such a CIRCLE appears in its environment of use. VEE is only useful in the MAN procedure, so is located there. Units such as those identified as NECK: and BODY: can also be chunked with boxes.

We caution the reader not to think of programming necessarily as an esoteric thing, as computers themselves were once considered. When programming becomes much more useful and incrementally learnable, it should, we hope, take its place beside written language as a complex skill, yet one which every literate person is expected to master to some appropriate level.

Figure 2 shows some familiar Logo-like programming activities that are carried out by typing commands into a kind of box we call a *do-it* box.

The social niche for a computational medium is much too large and complex to be described in any simple terms. In fact, it is misleading to think of it as a single niche at all, but rather we should think of it as occupying a large collection of niches. Indeed, a medium in our sense must

justify its existence by being able to play chameleon for multiple roles.³ We can usefully describe more specific social niches to first approximation as "genres," in analogy to various forms of written language. Accordingly, we are about to enumerate some new and (hopefully) important software genres that future learners will learn and recognize, taking care that we do not mistake superficial features of a genre for the network of social and personal activity structures in which the form works. Books may be pulp novels, which fit into a particular network of activities and interests, or they may be technical monographs, which fit into quite another. It will be our obligation to make such distinctions where appropriate.

Tools

Some scientists have tried to define human beings as tool-using animals. While contemporary comparative psychology does not allow such simplifications, it cannot be in doubt that important and special characteristics of humans and their civilizations can be seen in the tools they use. Especially relevant to us are technical and scientific cultures, which are rich not only in physical tools, from pliers to computers to particle accelerators, but they are also rich in intellectual tools. A scientist or a mathematician may use little more than a pencil, yet his life is full of "intellectual instruments" -- well-worn patterns of thinking that accomplish particular, frequently needed goals -- handed down from previous generations of his culture. In contrast, classrooms are substantially impoverished with respect to tools. Not only are they poor in physical resources, but children rarely come to the point that what they learn serves important functions for them in accomplishing what they wish to accomplish. Teachers are not much better off by these standards, though perhaps test booklets, mimeo sheets, blackboards and even "manipulatives" can count.

3. Here, common terminology fails us. One generally does not think of written language as a medium. Instead, its many physical manifestations are individually "media," such as books, magazines, newspapers, etc. We should probably be describing written language and our proposed computational medium as "abstract media," which have many physical manifestations.

We imagine, then, that a computational medium can provide the substrate to support a new tool-rich culture in schools. Children will learn by using, inspecting and even crafting tools to support activities they understand, and teachers will be much better supported for their own activities than at present. The transition to a culture that values its computational tools enough to spend substantial effort to support them has already happened in some corners of society -- and for secretaries and "just plain folk" as well as for programmers and scientists. The same can happen in schools.

With our emphasis on activity structures, we must say more about learning with tools. How might people learn, or fail to learn, with tools?

We can pass quickly through initial promises and expectations about tool use, which are widely extolled. Tools may give children and teachers capabilities in simple and concrete form that they need to engage in activities they want to engage in. In science, especially, tools can allow students to approach a different class of problems, say, more open ended and exploratory ones [cf. Robert Tinker's work]. It is said that tools can take over the menial tasks, say arithmetical or graphical ones (such as point plotting), to allow students to work at the more important higher levels of deciding on approach and interpreting results. Now let us enrich these near-platitudes by taking a first pass at how using tools may fail to engender learning because of inadequate attention to the activity structures in which tools are embedded.

It is actually quite easy to generate examples of how learning with tools may fail. Elementary school children's use of arithmetic is a good example. First, arithmetic can and perhaps should be viewed and used as a simple but powerful intellectual tool. Yet it is almost never learned in contexts in which it can be seen to be powerful. Instead, arithmetic is learned as an abstract practice, with the sole feedback to students on their capabilities at the *tool* level, rather than at the *use* level. Teachers say this sum is right, that multiplication is wrong. Not surprisingly, children do not build a real sense of number, or approximations or other parts of understanding arithmetic that connect to effective use in context. We discovered in early use of Logo that children's use

of number (in directives to the Logo turtle) was too fragile and failed them for the practical tasks they faced [Papert *et al.*, 1979]. They had no sense of a "little" or "a lot more," of scale or measurement capabilities of numbers. For example, students often considered 101 to be much bigger than 99; they chose numbers according to patterns -- 10, 11, 12 --- rather than according to meaning or need; and they frequently did not distinguish between distances and angles. At least some of this was quickly remediated in the turtle context, where numbers did things for the children.

There are other ways to see the problems of learning arithmetic out of use-context. Children may learn to operate the tool, but when arithmetic turns to "word problems," one finds that the tool is frustratingly difficult for students to use. It was learned in a way too disconnected from contexts of use.

Finally, even if children do learn to operate the tool in context, their understanding is likely to be fragile. The principles by which the tool works are never examined, alternatives are not understood, so children invent and perpetuate "bugs" in their procedures that they have no way of correcting on their own. Interesting corroboration of this mode of failure comes from research on how people understand calculators. Without inducing or being taught the underlying principles of operation of a calculator, but only teaching how to use it on some class of problems, users flounder in a sea of arbitrary button pressing when faced with even modestly different problems. In contrast, if users are taught or induce a model of the internal workings of the calculator, they can much more easily adapt the tool to new uses [Halasz & Moran, 1983].

At higher levels of instruction, no one questions that algebra is an important tool in learning, say, physics. Yet we see the tool mistaken for the deeper principles of the domain. One of the most prominent and well-documented types of failures in physics instruction is precisely of this sort. Students think of physics as learning equations, hence ignore the need for a semantics beneath the terms in the equation [diSessa, 1985]. In problem solving, students are not surprisingly stuck at the tool level and solve by exploring the combinatorics of terms in equations, rather than

exploring the more fundamental space of physics concepts as they apply to the problem [Larkin *et al.*, 1980].

Lillian McDermott [in diSessa *et al.*, to appear] provides a final example of a similar failure. In geometric optics, students are taught to use ray tracing and a few standard methods that work for a relatively broad class of problems. But, again, the tool seems to become the domain for students. The characteristics of the particular methods become the characteristics of optics, and one sees bizarre conclusions drawn about the world that are, in reality, reflections only of the tool provided to look onto the domain.

We can crystalize these reflections into an "ideal scenario" for the activity structure involved in deep and synergistic learning of tools and subject matter. In brief, we propose:

- Tools in context
- Understanding underlying principles
 - * Bottom up: Tools through design
 - * Top down: Inspectable tools

In more detail, tools should be introduced in a problem context that is, in some sense, thoroughly understood. In part this is to insure that the tools are clearly distinguished from the domain to be learned; in part this learning in context is to insure that selection and application principles of tool use are clear. To the extent possible, the features of the situation that motivate the tool and the approach to solution that the tool represents (and alternate approaches to the problem) should also be understood. In simplified cases, students should be able to carry out manually the tasks for which the tool is designed. This is aimed, in part, at making clear the principles by which the tools work, and to provide for alternative approaches when the tools themselves might fail. In the best case, students should build at least a simplified approximation to the tool, or specify the basic properties of the tool.

Where the bottom up approach, tools through tool design, is impossible, we would like to have tools built of pieces the student understands, and inspectable tools that allow students multiple partial views of their construction -- views that illuminate operating principles and use, or views that show (at an appropriate level of detail) the processes through which the tool accomplishes its work.

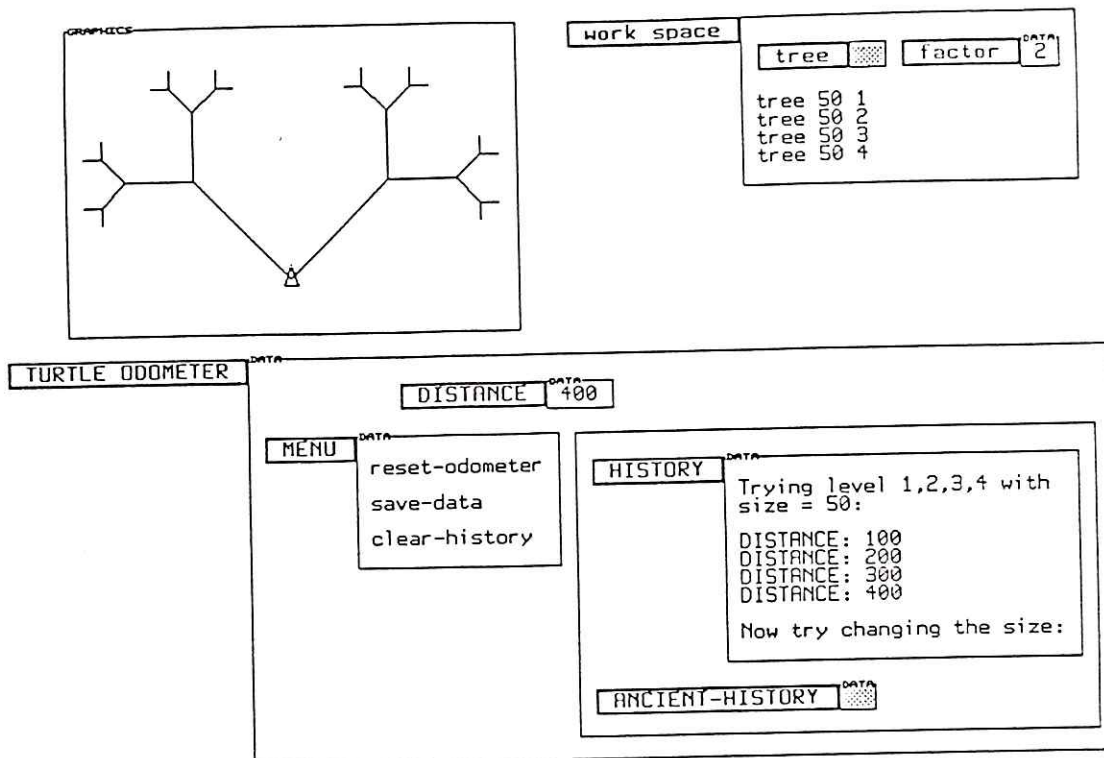
In a nutshell, students should be able to explore and hand-simulate tools; they should understand the usefulness and limits of tools in context; and they should be able to find alternatives when a tools fail. On a grander scale, we would like to encourage a community of teachers and students whose instincts are to build and share appropriate tools, not only in the clean and widely applicable style that we are used to seeing in scientific tools, but also for more everyday and particular tasks.

Returning to arithmetic, the literature shows that arithmetic *can* be taught according to these principles. Gaea Leinhardt [in press] documents a wonderful case of teaching a subtraction algorithm by thoroughly exercising the problem context, making sure students understand the resources they have (principles), and only after setting the scene, allowing students to invent the algorithm-solution. The inventive capability of students can be channeled through adequate preparation into deeper understanding and away from the prolific but unproductive invention of bugs.

Tools in Boxer

Figure 3 shows a very simple tool built in Boxer, a "turtle odometer," which keeps track of how far a turtle walks in its meanderings. Such a tool can be built from scratch in about 10 minutes by a competent Boxer programmer. The point-and-execute menu on the left is simply typed in place, and allows the student to reset the odometer or "dump" its value (as DISTANCE: ...) into a HISTORY box so as to keep track of data in experiments with the turtle. Here the student is investigating the length of a binary tree as the number of levels of branching is changed. The

(A)



(B)

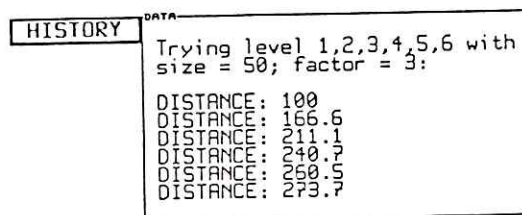


Figure 3. A turtle odometer. a. The total length of a tree diverges as number of branchings increases. b. The length of the tree converges in a different experiment.

tree has been programmed and is being executed from the box in the upper right of the screen. Of course, the student may type directly into the history as well. The menu item "clear-history" puts the current history into the box called "ancient history" providing a long-term log of activities. Capturing process in this way is extraordinarily easy in Boxer, and we expect it not only to ease and enhance use of tools generally, but as well to provide a better basis for reflection on the processes of investigation in addition to focusing on results. Note that the student has uncovered a very simple pattern in the length of the tree -- 100, 200, 300, 400 -- increasing

linearly with number of branchings. Figure 3b shows the quite different pattern when one changes the FACTOR (ratio of size of branch to stem) from 2 to 3. Now, the total length of the tree will converge with increasing level, never getting greater than 300. Another interesting investigation would be the lengths of spirals of various kinds (some diverge in length when they wind inward, some do not). Or, a surprise is in store for a student who checks the length of the path of one of four turtles situated at the four corners of a square and directed to walk constantly toward their clockwise neighbors.⁴

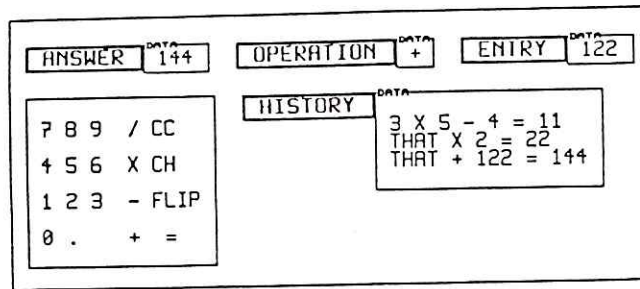
The turtle odometer illustrates some of the reasons a computational medium can contribute toward establishing real examples of our ideal scenario of learning with tools. A collection of tools in computational media can be easier to learn and more flexible at the same time since they are all similar, using the basic capabilities of the system to define how one interacts with and modifies them. A menu is just a list of commands waiting to be executed. Similarly, basic capabilities of the system such as text editing (e.g., typing directly into the history box, editing the menu) can serve instead of idiosyncratic special-purpose facilities.

Note as well that this tool is so simple as to be entirely under the control of the student. One is in very little danger of mistaking the tool for the task itself. Indeed, if the student does not build the tool herself, she can certainly inspect it and modify it if necessary (say, to adapt to the needs of the four turtles problem). Tools do not need to be complex to be useful and productive in a learning context, and we hope many small tools may contribute as much as a few complex ones to the tool-rich culture we envision.

More centrally a computational medium allows us incredible flexibility in crafting the context for learning tools and for learning with them. It is easy to generate and modify microworlds, like the turtle in this case. And these contexts can all be interfaced trivially to tools, as everything lives

4. It turns out the curved path of each turtle until it meets its neighbor is the same length as the sides of the original square, illustrating that the motion of the chased turtles, orthogonal as it is to the motion of the chasing turtle, does not affect the chase at all.

(A)



(B)

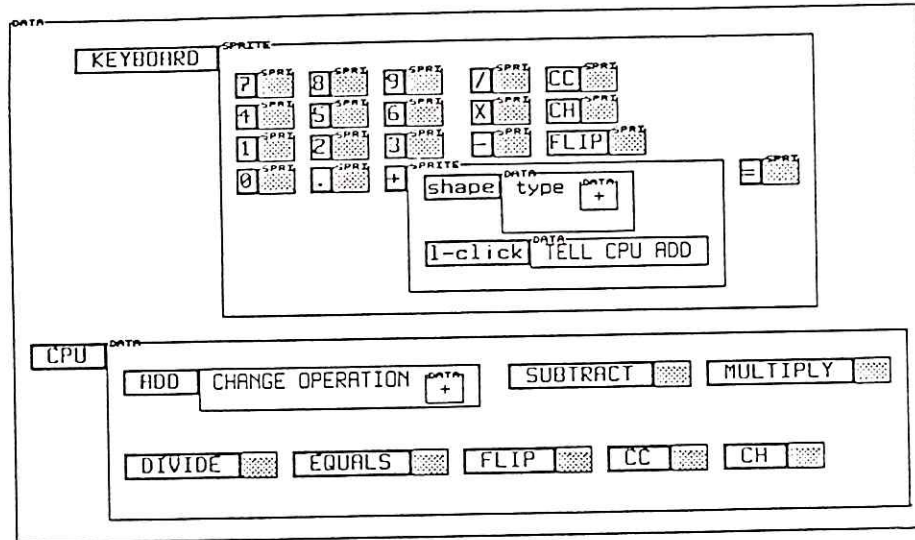


Figure 4. a. A Boxer screen calculator that works by clicking the mouse on the "keyboard." b. The calculator mechanism has many pieces, but each is very simple.

in the same environment. Indeed, a tool built for use with one microworld can easily be moved to another, in contrast to the extremely problematic interfacing of different applications (editor, database, spreadsheet, statistical package, programming language, ...) in the present state of computer use. Tools can even extend the microworld itself, by being programmed to act back on the microworld, say some features of turtle control can be added to the turtle odometer menu.

Figure 4 shows a screen calculator built in Boxer. This slightly more complex example emphasizes how Boxer's hierarchical inspectability can provide students with models of how the tool works by simply allowing its inspection through expanding and shrinking. Figure 4b shows

the insides of the calculator. We see boxes that define the individual keys of the calculator, their graphical appearance (SHAPE), and their responses to mouse clicks (L-CLICK). We also see the central processing unit (CPU). Inside the CPU are the various commands that manipulate the "registers" on the surface of the calculator, ANSWER, ENTRY, OPERATION and HISTORY. Any piece of the calculator can be executed with a point and click to see how it works in isolation.

Like the calculator, any Boxer tool provides a model of itself, and if we take some care, these models can be nearly as informative as we wish. Again, modifications and extensions can be added by students and teachers to extend the usefulness of the tool beyond the range for which it was originally intended.

In our examples, we have slighted the importance of tool-use for teachers. However, at this stage it should be easy to imagine how Boxer can be used for teachers. In the first instance, learning tools are quite relevant for teacher training, prior to and during service. But I also think more mundane tools will be perceived as important by teachers, and will provide important learnability through incremental usefulness. Simply to keep track of and organize notes, plans and grades in a convenient hierarchically organized "book" is substantial help. One can add simple tools to, say, warn the teacher of students doing poorly (at least those getting poor grades), or automatically to grade and store results from quizzes. Indeed, it is relatively easy to make programs to generate simple quizzes, perhaps tuned to students' levels of achievement. Making or modifying pre-made "interactive work sheets" for students would also be, we project, common activities for teachers. We have produced examples of all of these genres, though classroom use by teacher is yet to come.

The Boxer group has some preliminary empirical results with students learning in this tool-oriented manner. Ploger & diSessa [1987] show a 12 year old student in his first hour and a half contact with Boxer learning some central ideas having to do with probability and statistics by using and building simple Boxer tools. We were extraordinarily pleased to see how well this

scenario matched our "ideal scenario" of learning tools in context, from the bottom up (through design) and from the top down through inspecting and modifying supplied "transparent tools."

Knowledge Spaces

Boxer's hypertext capabilities augmented by programming provide an opportunity to represent knowledge externally in ways that until now have been inaccessible to most humans. The point is simply that with a much more expressive medium, extended beyond the linear and static representational capabilities of text, we may be able to present, so as to be effectively and reflectively accessible, much more of the complex weavings of relations and possibilities inherent in the knowledge of some domain. I call such presentations *knowledge spaces*. So, for example, biology might to some seem to be simply a swarm of facts and definitions. Yet with an appropriate presentation, we could highlight relations more than isolated facts, or highlight large-scale organizational principles. In another model, we might aim to take the burden of fact recall at least partially away from students, so as to support focus on the use of that knowledge in problem solving, and on problem solving strategies themselves. Again, it is facile to pretend one can simply lay out the knowledge of some domain for inspection and appropriation, but if we pay attention to the structure of activities built onto such displays, new learning patterns may be created.

An early model in my own thinking about knowledge spaces occurred several years ago when my then 9 year old son demanded that I teach him about algebra. Without having time to think out a curriculum, the two of us sat on the couch with a pad and pencil, and I started asking him about how much he already knew. It turned out that the idea of attaching a "variable" number to a symbol was in his repertoire. So I began trying to see if the axioms of algebra made any sense to him. Did he know, for example, that $A \times B = B \times A$. Yes, indeed, that seemed right to him. When I asked how he knew, he announced that $A \times B$ was the area of a rectangle, and if you turned in on end, that would not change the area of the rectangle, but you would write it $B \times A$.

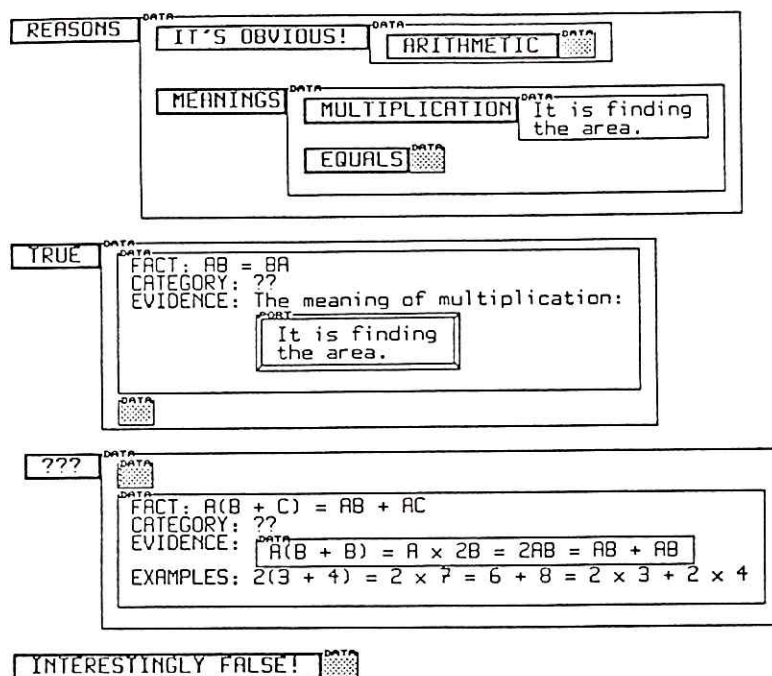


Figure 5. A hypothetical knowledge space on algebra constructed (mostly) by children trying to understand the domain.

The distributive law, $A \times (B + C) = A \times B + A \times C$, was more problematic. At first he had no idea if it was right or not. But we tried a few examples, and it seemed to work. Although he considered this clear evidence, he did not consider it definitive proof. After a bit more work, he announced spontaneously that if B and C were the same, then he could see how it worked. $A \times (B + B) = A \times 2B = 2 \times AB = A \times B + A \times B$. Apparently having collapsed $B + C$ into one term, $2B$, and thus eliminating the "distribution," he could see reason behind the law.

I won't continue the recounting, but would like to build this small episode into a plausibility case for the following activity. Imagine a group of children at a relatively early stage in trying to understand some domain like algebra. Let us put to them the task of organizing all they know and their reasons for believing those things into a knowledge space. We should, of course, provide some examples and some schematic structure on which to hang their developing ideas.

Figure 5 incorporates slightly idealized bits from the above scenario into a partially built knowledge space. Note that we have made room for developing knowledge and even the history of idea development in categories like TRUE, ???, and INTERESTINGLY FALSE!, say, $1/(a + b) = 1/a + 1/b$. Each true item needs evidence or reasons, possibly from examples, or possibly from categories of reliable knowledge, like geometry facts such as $A \times B$ is the area of a rectangle. In the figure we have connected pools of reliable knowledge to the items they support using ports. The boxes containing those pools reify important epistemological categories, while ports permit reasons also to be distributed where they are needed in the logic of the subject. The evolution of reliable pools, possibly seeded concretely but subtly by teacher insertions in the knowledge space, could play an important role in the evolution of the presentation. Could some problematic "fact" be understood in the same way some previous fact was understood -- say, is there a geometric presentation of the distributive law? Try it!

Perhaps it is entirely fanciful to imagine that students would build something like an axiomatic deployment of some domain in this way. But I don't think it is fanciful that they would learn a lot about the domain, what is true and false, why and how one can organize and think about the whole system. I have no doubt such work could at least contribute to the appreciation of the difficulty of achieving such a compact representation of a field as an axiomatic deployment, or the possibility of multiple such organizations as opposed to the presumption built into most mathematical presentations that there is only one way of thinking of a domain (a presumption, incidentally, that students seem to learn all too well).

There are several reasons why I imagine this as an activity of a group of students over an extended period of time. First, it is a difficult task, and can well absorb the insights of a group of children. I am quite certain an individual child would soon stall for lack of ideas. In addition, I imagine that the community effort involved in arguing about and negotiating organization for the community knowledge space would prompt a lot of reflection and learning that an individual child would not meet even if he were capable of completing the task.

There is much to learn before implementing a knowledge space "curriculum" (sequence of activities). How much teacher intervention and seeding of the knowledge space should be done? Which seeds in various domains will actually flower in children's hands? What is the appropriate relative emphasis on constructing, reorganizing and reflecting? We intend to explore these issues as part of a project to teach elementary school children about the topic of motion.

Another model for knowledge spaces is the work of a graduate student, Steve Adams. He is studying the knowledge and capability of child dinosaur experts (between ages 9 and 12) for the purpose of enhancing the learning of even these children, and figuring out how to engage children who would not ordinarily come to such expertise. It turns out such child experts have a surprisingly large range to their expertise and interest, certainly including an extraordinary range of facts, but driven by interest and competence at multiple levels of knowledge, from "empirical generalizations" ("It seems to me most two-legged dinosaurs are meat eaters.") to sophistication and interest in the progress of science ("I wouldn't buy a book with a copyright before 1986; things are changing too fast." "What's really interesting is when scientists get things wrong, like the apatosaurus ...").

At the core of the microworld Adams is building in Boxer is a more or less conventional database (Figure 6). This, of course, allows some usual database activities like querying to find or verify correlations such as the two-legs/meat-eating one mentioned above. But since it is in the midst of a computational medium, all sorts of other activities can be envisioned. Indeed it is the design and support of these activities that makes the database really interesting. *The transition from database to knowledge space rests on the medium providing support for specific activities in which children will learn.*

Perhaps the simplest activity is to peruse and add to the database. This may be done, of course, using only basic Boxer editing capabilities. Provision has been made to encourage introducing items of special interest, for example, in a STORIES field in the entry template. Another activity, which was used as a knowledge assessment instrument, proved surprisingly engaging

allosaurus

apatosaurus

DATA

Name

apatosaurus

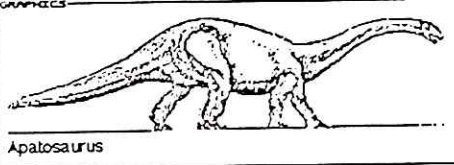
Pronunciation

ah-PAT-uh-sawr-us

Meaning

"deceptive lizard"

Picture



Number-of-legs

4

Length

75 feet

Weight

30 tons

Food

plant-eater

Fossil-Locations

Western North America Europe

Classification

sauropoda sauropodomorpha saurischia

Stories

DATA

The names "Apatosaurus" and "Brontosaurus" refer to the same animal. Fossil bones discovered in 1877 were given the name "Apatosaurus." They are the same as bones discovered in 1899 and named "Brontosaurus."

compsognathus

ceratosaurus

diplodocus

tyrannosaurus

archaeopteryx

Figure 6. A piece of a dinosaur database, the activity center of a dinosaur microworld.

and productive. That is to take the entries of the database and re-organize them according to any principle the child wishes. One could sort according to appearance, according to geological period, according to how fast they could run (a suggestion of one of the child experts, who then proceeded to invent a theory of how to tell how fast a dinosaur could run). Adams has seeded other activities with simple programs, suggesting ways to go. One is a quiz program that is easily extended to different classes of questions. Students can invent a class of questions and then build a question and answer key. The key itself may be literally the result of a simple query to the database. (The query provides the answer to the question for each dinosaur.) Students should learn in inventing their own classes of questions, and in taking their own and other students' quizzes. We know that this is highly motivating for students from previous work with Logo. Boxer's inherent database capability makes such activities easier to begin (e.g., perusing

- 21 -

and directly modifying), more broadly engaging (e.g., including pictures) and more extensible (through better inspectability of existing program and simpler programming more generally).

I will close this section with an example of children's spontaneous activity that highlights one of the special features of Adams' designs, a feature that challenges most views of the instructional designer's task. It also challenges our scientific competence to understand activity structures, and even our common instincts for the direction of scientific work in instruction and learning. Two of Adams' dinosaur experts took to drawing cartoons for each other based on corruptions of real dinosaur names. So, for example, a dimetrodon became a dime-meter-odon (Figure 7), with a dime slot for a mouth and a meter needle on its fin.

A frenzy of activity of this sort over a several hour period resulted in dozens of such cartoons. It is difficult to say exactly what is learned in such an activity, though some after-the-fact pronouncements might be made. Certainly this would be among the better ways to generate mnemonics for dinosaur names, and to become acquainted with some species partners know but you don't. But the point is precisely that such narrow, cognitive views of judging the value of activities may be beside the point. Instead, what seems more important is that the activity provides an opportunity for enjoyment based directly on their expertise, and an opportunity to share ideas, factual or whimsical. I believe such activities played a central role in what made these children dinosaur experts in the first place, almost independent of particular knowledge outcome.

I suggest our emphasis on activity structures should add a new direction to a constructivist view of learning: Not only must we design for continuity of knowledge, but we must design for *continuity of goals* as well. What we expect as personal goal-directed engagement should lead only gradually from what learners initially consider interesting and meaningful enterprises toward what we set as educational targets, say, acting in the world as little scientists and mathematicians. Designers should be chastened that without due consideration to activity

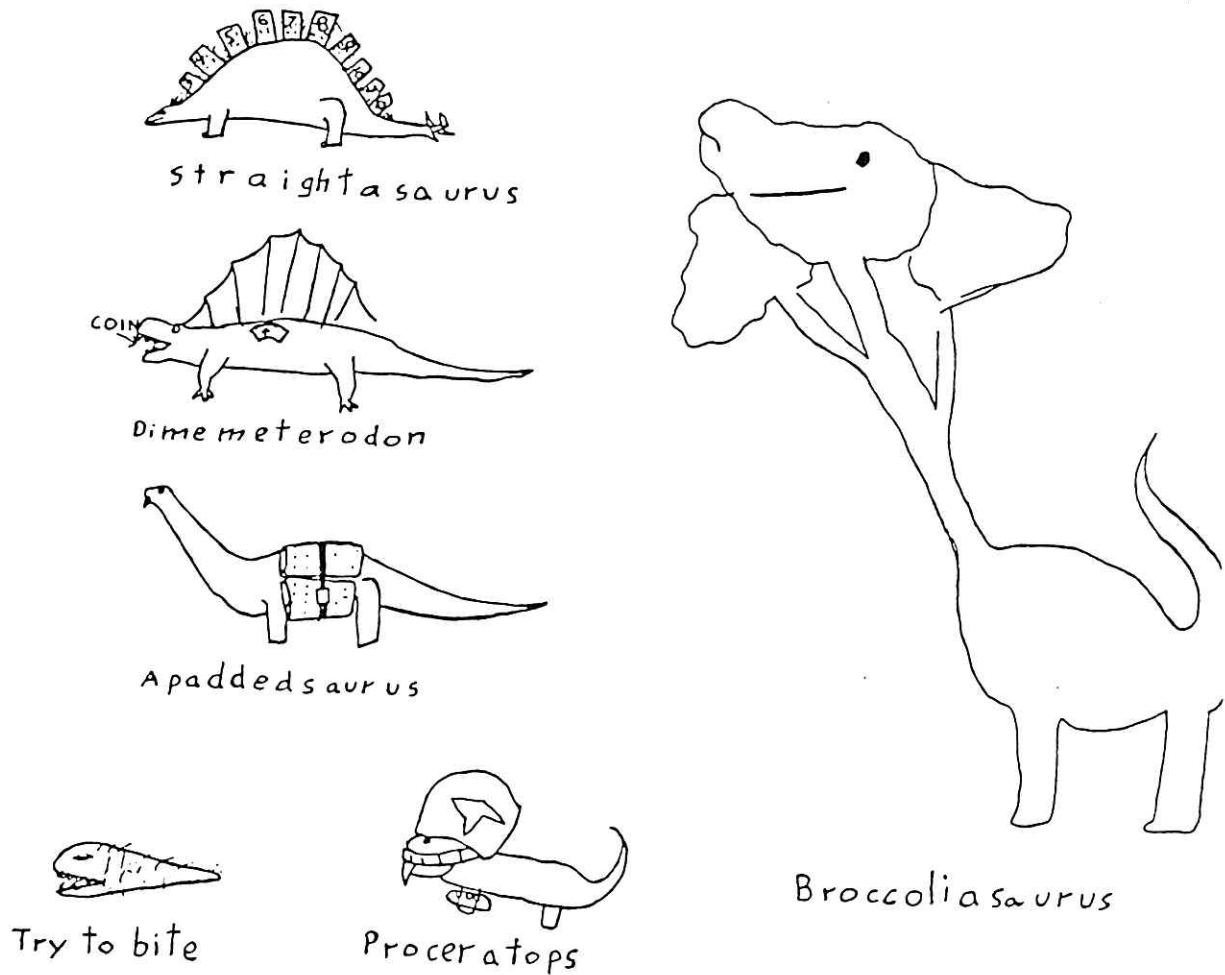


Figure 7. A spontaneous social activity, trading dinosaur cartoons: straight-osaurus (stegosaurus); dime-meter-odon (dimetrodon); a-padded-saurus (apatosaurus); try-to-bite (trilobite); pro-ceratops (protoceratops); broccoli-saurus (brachiosaurus).

structures and continuity of goals, they might never consider dinosaur cartoons a legitimate activity, or that they might design such activity *out* of instruction by default.

Mind Toys

Children and adults engage in word and language play, from babbling to crossword puzzles, and are almost always better off for having done so. Any respectable medium should have such possibilities. In the case of computational media, such play can serve double duty in gaining

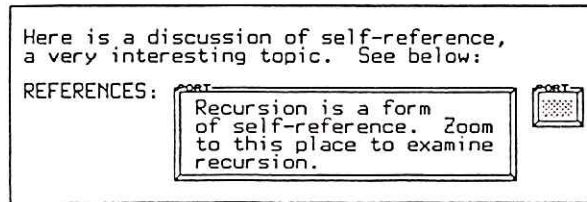


Figure 8. An item on self-reference with port cross-reference to recursion.

familiarity with and power over the media, but also, since the medium itself has mathematical-structural properties, it can be play with mathematical and scientific ideas at the same time.

The Logo community has witnessed an interesting social phenomenon. If one looks through books and newsletters on Logo, and listens at conferences, one finds a tremendous fascination with the topic of recursion (programs that use themselves as subprocedures). Recursive designs, in fractal, space-filling and other forms, occupy an impressive amount of attention, as does the topic of understanding the recursive programs that generate them. In the abstract, it is hard to imagine that such an exotic and specialized mathematical topic should become a central concern in any non-mathematically-sophisticated population, let alone among school teachers and children. Yet this is the case. The intrinsic interest in this structure, which creates elaborate patterns seemingly out of thin air, coupled with the fact that simple programming puts one in the position of experimenting with it, has nearly made a popular craze out of it.

There are even simpler structures in Boxer that may play the role of toys for the mind for future generations of computationally literate people. As mentioned earlier, ports make excellent cross-references to related material in a Boxer world. But what happens in the case of self-reference? What happens when a reference in a chunk of Boxer references the chunk itself? Figure 8 shows a box that contains two references to "further material." The first is an ordinary reference, but the second is a reference to the object that is referring. That is, the second port is a port to the

box containing the port. Before looking at Figure 9 (at the end of the paper), try to imagine what will happen if one merely shrinks the first port and expands the second.

Behind this almost familiar effect (which one sees on the evening news when the monitor behind the anchor contains the feed from the camera, hence an image of the monitor, which contains ...) is a deep mathematical idea: Any structure that contains itself (say, in a port), plus some other stuff, *must* be infinite.⁵ Infinity is an always intriguing phenomenon, especially when it seems so near to us, having been created simply with "everyday" objects and operations -- a bit of mystery and simplicity in an elusive but nearly-graspable package. Part of the root attraction of this phenomenon is that our common sense says it cannot happen. It is the same common sense that led Kant to claim that the whole *must* be bigger than the part. Yet, here it is, a part that *is* the whole, and vice versa -- made with a few keystrokes.

Designing for rare events

I do not claim that toying with such a phenomenon, by itself, leads to deep mathematical understanding. Though I believe there are many such experiences, (perhaps not so spectacular as this, but which may be just as personally meaningful) lying in wait in a computational medium, I think the importance of such experiences goes beyond their cumulative direct effect in developing mathematical sophistication. I am looking to rare but profound experiences that children may have with something that they can see as mathematical (or merely intellectual), that can change their minds about what is interesting and exciting to do. I recall, myself, a handful of such experiences in my childhood that, in retrospect, I believe were important to my entering a scientific career. I would conjecture that 7 plus or minus 2 profound experiences of this sort might entirely turn around a student's perceptions of what she wants to spend time on. Of course, such a conjecture is far beyond our capability to prove or even genuinely to understand at this stage of the scientific understanding of learning. But, again, I wish to point to patterns of

5. This corresponds to the standard Dedekind definition for infinity, a set that contains itself as a proper subset.

activity, this time the relation between possibly rare but important events and life long learning patterns, as deserving more attention than we give them. Designing learning environments for profound but rare events may be more than we can imagine carrying out competently at this stage. But designing so as to exclude them by default may be more dangerous than we should bear.

Summary

The central objects of our concern in the design of educational artifacts ought not to be the technology itself, but instead, the specific activities of students and teachers using the technology, and the social niches into which these activities fall. If we do not realize this, we are in danger of perpetuating old and ineffective patterns or worse, at the same time that we introduce dazzling new machines. In this context, I have examined ways we can most profitably use coming powerful extensions of written text, computational media.

I have argued that tools can be sterile objects unless we provide for appropriate activities with them, including designing them and using them in contexts that exercise *and* measure the tool's power. On the other hand, new computational media can support such activities in ways that to now have been inaccessible, providing better learnability, flexibility and extendibility. There is hope schools may be substantially changed into tool-rich cultures that simultaneously empower and enlighten. I have argued that the extended representational capabilities of computational media provide important new opportunities, such as knowledge spaces, in which to present and reflect on the nature of a field and on one's own development of knowledge. Again, this can only happen if we investigate and design appropriate activity structures to properly connect the student's mental life to what appears on the computer screen. Finally, I have argued for both the importance of rare events that may play a substantial role in intellectual development, but also for the importance of concern for the continuity of goals and activity structures in their own terms.

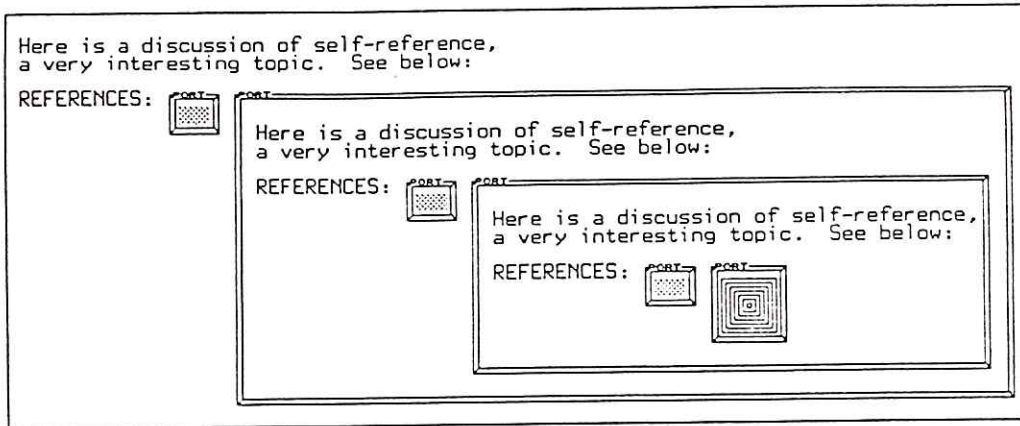


Figure 9. The item references itself, leading to an infinite structure.

Acknowledgements

I wish to thank the following for helpful comments on drafts of this paper: Steve Adams, Melinda diSessa, Don Ploger and Alan Schoenfeld.

References

- diSessa, A. (1985). Learning about knowing. In E. Klein (Ed.), *Children and computers*, San Francisco: Jossey-Bass Inc.
- diSessa, A., Gardner, M., Greeno, J., Schoenfeld, A. & Stage, E. (to appear). *Towards a scientific practice of science education*. Hillsdale, NJ: Lawrence Erlbaum.
- Halasz, F. G. & Moran, T. P. (1983). Mental models and problem solving in using a calculator. *Proceedings of CHI '83, Boston*.
- Larkin, J. McDermott, J., Simon, P. & Simon, H. (1980). Expert and novice performance in solving physics problems. *Science*, 208(20), 1335-1342.
- Leinhardt, G. (in press). The development of an expert explanation: An analysis of a sequence of subtraction lessons. *Cognition and Instruction*.
- Papert, S., Watt, D., diSessa, A. & Weir, S. (1979). Final report of the Brookline Logo Project. Memo 545, Cambridge, MA: M.I.T. A.I. Laboratory.
- Ploger, D. & diSessa, A. A. (1987). Rolling dice: Exploring probability in the Boxer computer environment. Boxer Technical Report E1, Berkeley, CA: School of Education, University of California.