12. The Medium and the Curriculum: Reflections on Transparent Tools and Tacit Mathematics

Chris Hancock

TERC, 2067 Massachusetts Ave., Cambridge, MA 02140 USA

Abstract. This paper explores the phenomenological and curricular dynamics of implicit mathematical structures embodied in "transparent" computer-based tools. Examples from a clinical study of students working with the *Tabletop*™ database/ data analysis environment illustrate the process by which disruptions of transparency can provoke increasingly reflective use of a tool and bring students into engagement with valuable mathematical ideas. The interaction among learner, medium and curriculum is seen to have important implications for pedagogy, tool design, and evolving conceptions of mathematics.

12.1 Introduction

Computer-based tools figure increasingly prominently in visions of curricular reform. Their ability to capture, display, store, transmit and transform representations of physical and social phenomena and of human communication offers the hope of empowering children to reach beyond isolated subject matter and the walls of the classroom into a wider and more meaningful world. The capabilities of tools—data analysis, information access, scientific data gathering, video analysis, dynamic modeling, etc.—sometimes attract more attention than the tool structures which underlie these capabilities. The representations and operations of a computer tool constitute a mathematical structure, a model implicit in the tool. This paper is concerned with the status of this underlying mathematics: in students' experience as they learn to use computer tools, and in evolving conceptions of mathematics curriculum.

Many people (novices and non-experts) use computer tools without being much aware of the workings of the tools' implicit models, or even of their existence. Their use of a tool is guided instead by their knowledge of the situation being modeled. This is a kind of naive realism in which the mediating mathematics is invisible, at least for a while. We value tools whose design makes them conducive to this kind of naive realism: We may call them "transparent." Much effort goes into

trying to build intuitively accessible representations and fluid, flexible ways of acting on those representations, in the hope that, on a clear day, a student might look into a computer and see through the glass screen, through the tool interface, through the mathematical model, all the way to a real world situation of interest to her or him— and to have this transparency endure for a substantial period of interaction.

Because the model is only a model, there must inevitably be an exodus from this computing Eden. The transparency of the interface breaks down, and the computer tool becomes clearly and frustratingly distinct from the "object of interest" to which it had previously given unimpeded access. Such moments can be important in a child's experience. Episodes of lost transparency can become insurmountable obstacles that cut short the child's engagement, or minor intrusions of inexplicability that loosen but do not break the child's grasp of the activity. More positively, they can be opportunities to learn more about the medium itself—the tacit mathematics of the computer tool. Such episodes also raise important questions for the educator concerned with software and curriculum design. Should the breakdown be interpreted as a failure of the environment to live up to its goal of transparency, or as a natural and valuable component of students' engagement with important mathematical ideas? The answer is different each time, and involves a judgement about the learnability and cultural relevance—in other words, the curricular appropriateness—of a particular piece of mathematics as it is embodied in the tool.

Why work to create transparency if we know it will be broken? Modeling tools with a capacity for transparency can help us experience and understand mathematics as a way of seeing the world. Mathematics itself becomes not just something to look at, but something to look through. The tool's mathematical model and the object to which it is applied can illuminate each other, through both their similarities and their mismatches. Like glass and mirrors, some mathematical domains can best be seen by looking at the distortions and effects they create when looked through.

12.2 Transparency and Tabletop

Let us dig a bit deeper with the help of some specific examples of students grappling with a disruption of transparency within a computer environment. These examples come from a small study conducted by Aaron Falbel and myself (Falbel and Hancock, 1993). The study used a prototype version of the *Tabletop* database/data analysis environment, developed at TERC. While much of what we saw appears relevant to most data tools, a few facts about Tabletop will be helpful in understanding what happened.

12.2.1 Tabletop Essentials

Tabletop is a tool with which students can construct, explore and analyze simple (i.e., record-oriented) databases. It was developed to support a curricular move-

ment in the United States towards increased treatment of data analysis and statistics as subjects of study within mathematics, and increased use of data and data analysis to support projects and inquiry in many subjects. The illustrations in Figures 1 through 5 provide a visual overview of some of Tabletop's capabilities at the time of the study (a more powerful version has since been developed for publication).[1]

name	area	popula-tion	continent	gnp per capita	labor force: agriculture	phones per 1000	prin lan
Australia	2968	14.6	Australia	11206	6.5	355	Eng
Austria	32	7.5	Europe	8756	1.5	245	Ger
Belgium	12	9.9	Europe	9918	3.0	257	Fre
Canada	3852	24.0	N. America	11026	5.5	527	Eng
Denmark	17	5.1	Europe	11116	8.3	408	Dar
Finland	130	4.8	Europe	9886	11.5	329	Finr
France	210	53.7	Europe	10637	8.7	217	Fre
Greece	51	9.6	Europe	3992	5.1	207	Gr
Iceland	40	.2	Europe	13876	15.6	387	Icelar
Ireland	27	3.4	Europe	4762	19.5	121	Eng
Israel	8	3.8	Asia	5329	6.3	214	Heb
Italy	116	57.0	Europe	5155	14.2	230	Ita
Japan	144	116.9	Asia	9372	10.4	357	Japan
Luxemburg	1	.4	Europe	14231	5.4	384	Fre
Netherlands	16	14.1	Europe	10055	4.9	321	Du
New Zealand	104	3.1	Australia	7591	11.0	488	Eng
Norway	125	4.1	Europe	13463	8.4	330	Norw
Portugal	36	9.9	Europe	2282	22.3	111	Portu
Spain	195	37.4	Europe	4894	19.0	182	Spar
Sweden	174	8.3	Europe	13536	5.6	612	Swed

Figure 1. The Tabletop database/data analysis program provides a conventional row-and-column view for entering and editing data.

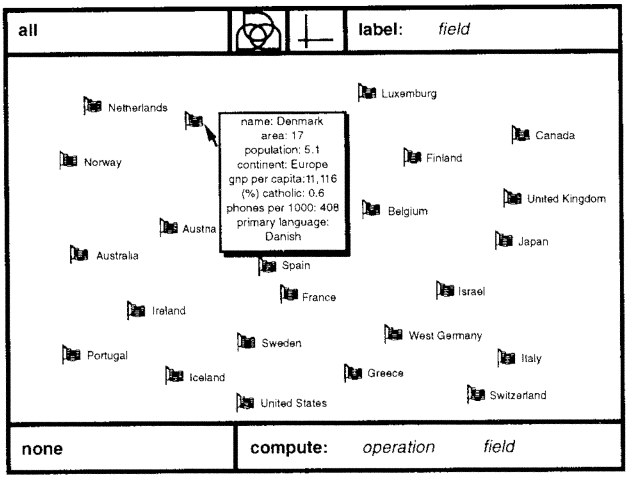

Figure 2. The "Tabletop window" shows one icon for each item in the database. The icons are initially scattered randomly. Any icon's detailed information can be examined at any time by pointing and clicking the mouse. Icons can also carry labels with information from any field of the database (in this case, country name).

[1]Tabletop and Tabletop Junior, for Macintosh or Windows, are distributed by Broderbund Software, Novato, CA USA.

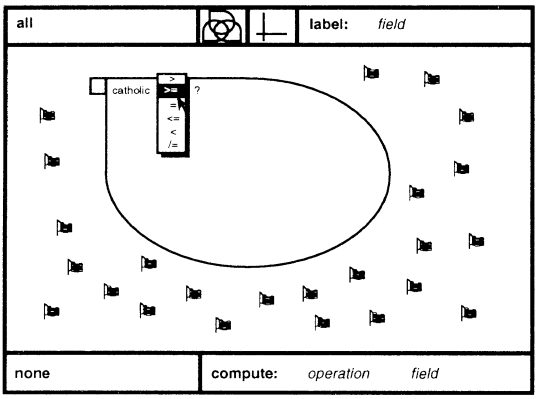

Figure 3. A set constraint can be established by using pop-up menus to select a field and a comparison operator, and by entering a comparison value. Icons satisfying the constraint move into the circle. They move quickly and simultaneously, but smoothly so that any individual can be tracked.

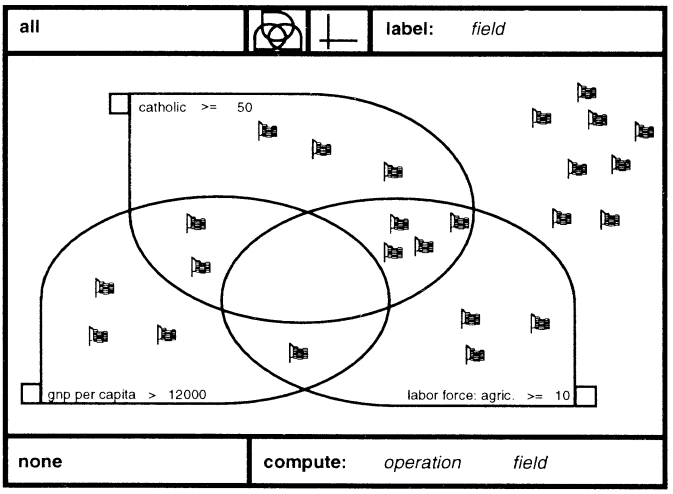

Figure 4. Up to three circles can be active at once. Any part of any constraint can be modified directly, and the affected icons will immediately move to new positions.

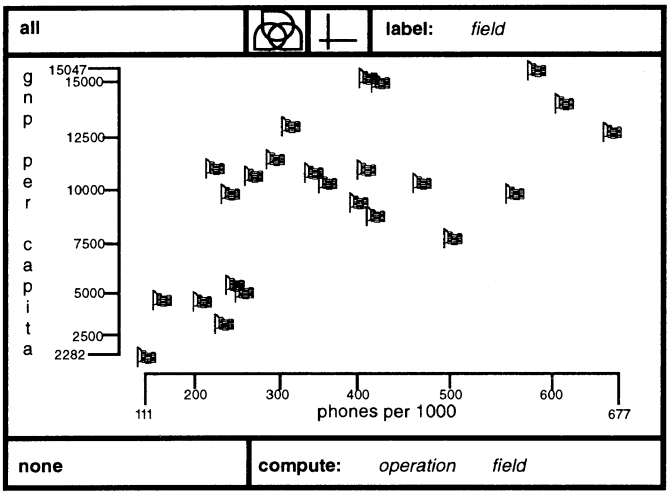

Figure 5. Through various spatial arrangements of icons, combined with optional summary computations, Tabletop can generate frequency distribution graphs, scatter plots (shown here), crosstabulations, and other plots. (Tabletop Software and interface design ©1989-1994, TERC, Inc.)

Tabletop offers two coordinated representations of a database: the row/column window and the "Tabletop" window. The row/column window is used for database construction and data entry. Information in the row/column window is arranged in rows and columns, corresponding to database records and fields. The Tabletop window presents the database as a set of small animated icons. These icons can be arranged automatically into Venn diagrams, scatter plots and other arrangements useful for data analysis. The appearance of an icon is editable, so that in a database of cats, for example, the icons can look like cats, and one can think of Venn diagrams and scatter plots as though one were actually arranging the real cats. When students make databases about themselves and their classmates, they routinely talk about Tabletop icons as though they are people: "Paula and Nathan went in here" "How come I went above Mary?" etc. This illustrates, in a modest way, one kind of transparency in a mathematically based rendition of the world.

The Venn diagram, with which we are concerned here, is a spatial alternative to the traditional representation of a database search query. It can have between one and three overlapping loops. In each loop the user may specify a set constraint in three parts: a field, specified by choosing from a menu of all the fields in the database; a comparison operator (=, >, <, etc.), again chosen from a menu; and a value, which can be typed in. Objects for which the mathematical statement so specified is true will slide automatically into the loop and all others will slide out.

12.2.2 Subjects and Method

Our study originated in the context of a project in which we worked with teachers in an urban elementary school to carry out extended piloting of Tabletop in a classroom context over a period of two years. Students in grades 5 through 8 (that is, aged 10-14) carried out a variety of data collection and analysis projects, including an "All about Us" activity examining assorted personal trivia; an investigation of neighborhood recycling patterns; a Coke/Pepsi taste test; a nutrition study comparing the calories, carbohydrates, protein, etc., consumed on one day by 60 students; consumer information about sneakers; and others. All students had experience adding data to a database, and most of them had experience creating databases of their own and knew how to add fields as well as records. Although students were generally confident and successful when creating databases of their own, we decided to look more closely at students' ability to decide when to enter new data and how to organize it. The students were accustomed to making databases (often with the teacher's guidance) and then seeing what graphs they could make. The problem of envisioning a desired graph and then figuring out what kind of data to enter would be a reversal of the usual process, and one that might require some challenging backwards reasoning. We devised the following task, which we call the "group separation problem," as a distillation of this issue.

To set the problem, a database consisting of a single column of names is created in the subject's presence. Some are male names and some are female names.

Name
John
Mary
Sally
Tim
Betty
Robert
Sue

Figure 6. Initial database for the group separation task.

Now a simple problem is posed. Can the subject get the computer to place the icons representing the male names in one Venn ring and those representing the female names in another. The question is a bit tricky in that the problem cannot be solved without returning to the database window and adding additional information. Ordinarily this information would be in the form of a new field with the heading "sex" or "gender" or "boy/girl" or some other such designator and an entry corresponding to each name specifying whether the person is "male" or "female," "boy" or "girl" (less conventional solutions are also possible, as we discovered). Once this is done, two loops can be made in Tabletop window with constraints referring to the new field. For example, adding a gender field as in Figure 7(A) makes it possible to create a Venn diagram as in Figure 7(B).

(A)

Name	Gender
John	male
Mary	female
Sally	female
Tim	male
Betty	female
Robert	male
Sue	female

(B)

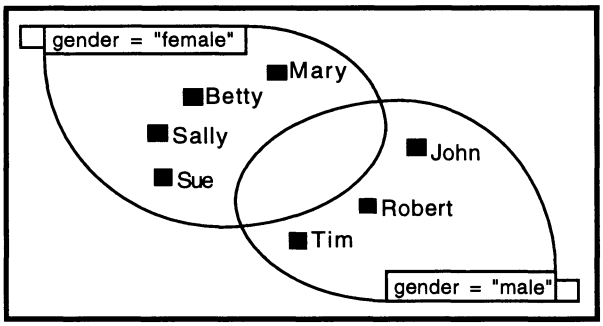

Figure 7. Sample solution to the group separation problem.

We posed this task individually to thirteen students, aged 10 to 12, who each had between one and two academic years' worth of familiarity with Tabletop. We also used follow-up problems with a similar structure: separating cats from dogs, married people from single, children from adults. The problem appeared to require a kind of interaction with Tabletop that was new in their experience, leading to the problematic intrusion of layers of the interface which had been previously transparent to them.

A sense of this previous transparency can be gleaned from the way that many students began the task. They proceeded directly and with apparent confidence to build a Venn diagram. Beginning at the first slot in the constraint, they found that the single available choice was `name` (corresponding to the only field in the database). Here I had the impression of a surprised hesitation. When we asked students what was wrong at this point, they were not able to say; nevertheless, I believe that if the word "gender" had appeared as an option for the first slot of the constraint, they would have used it without hesitation. Indeed, more than one previous database in class projects had had a gender field, which students had used successfully in constructing graphs and Venn diagrams. Since field names had always been ready to hand when needed, students had not previously had to question how they came to be there.

Eventually, with or without hints from us, all subjects realized that they needed to input data into the row/column window, to specify who was a boy and who a girl. We had expected to be focusing our attention on whether and how students came to

realize that the computer needed to be told who was a boy and who a girl, since from the subject's point of view it is obvious from the names. While many students did take time to come to this realization, the part of the task that proved far more challenging to them—and more interesting to us—was the process of figuring out how to get this information into the computer in a usable form. This was a hard problem for most of these students; only two solved it without substantial hints and guidance. We begin with one of the two successful ones, atypical in many ways but potentially illuminating for the topic at hand.

Example #1: Transparency of Words (The Case of Kamil)

Kamil (age 12) first tries to solve the boys/girls problem within the Tabletop window. He tries a Venn loop with `Name="girls"`. Then he clicks twice on the field section of the constraint. The only field available is `Name`. "I'm trying to change this," he comments. Then he asks: "Does the computer know that they are girls?" No, we reply. "Then you can't do this." He double-clicks on an icon, bringing up a small window with the person's name and no other information. "I need to tell it more," he says. Still, for a while, he keeps trying variations in the Venn diagram. He tries `Name > "boys"` and all the icons go in. (With textual data the software interprets less than and greater than in terms of alphabetical order, a feature which we have learned is more confusing than useful.) "The computer thinks they're all boys." We ask: Is that because of this symbol (the greater-than sign)? "That means that there are more boys than girls."

Quite soon he decides that the computer needs to be informed who is a boy and who is a girl. He adds a new field, called `Boys`, but he stops before entering any data and asks, "Does the computer know what 'boy' means?" We answer, "Well, it knows that as well as it knows what 'John' means." He does not find this reassuring, and after a few moments of puzzling, he says, "Wait—I could put Calories! Boys eat more calories than girls." (His class had done a study the previous month in which each student had counted their calories for a day, and the class had used the resulting data to try to conclude who as a group ate more calories, boys or girls. There had been a slight trend that boys ate more than girls, but there was a lot of overlap between the two groups.) Kamil then removes his empty `Boys` field, and adds a new field called `Calories`. He assigns each boy 202 Calories, and each girl 100 (Figure 8A). Then he builds a Venn diagram with one circle labeled `Calories < 200` and one labeled `Calories > 200`. Sure enough, the girls go into one circle and the boys into another (Figure 8B). We ask, "But what if a girl really ate more than 300 calories?" "You could take it down, he says." "Would that be lying?" "Well you could make the boys bigger."

(A)

Name	Calories
John	202
Mary	100
Sally	100
Tim	202
Betty	100
Robert	202
Sue	100

(B)

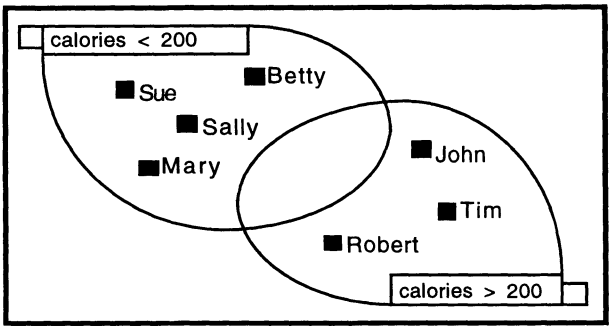

Figure 8. Kamil's first solution to the group separation problem, based on a fictional `Calories` field.

Kamil says that this is easy now that he has the hang of it. He says he could use the same technique on other problems, and he does. Given seven slips of paper with cats and dogs, he makes a database with an `Animals` field containing the animals' names, and a `Pounds` field in which each dog is given 102 and each cat 90. Given seven people to separate by marital status, he creates a `Years` field in which he enters 10 for single people and 20 for married people. Both of these solutions work. In both cases he also cautions, "This isn't going to make any sense," and prefers not to discuss any rationale or metaphoric basis for his choice of field names and values.

To see if Kamil's roundabout solutions reflect better knowledge of numerical than categorical data, one of the researchers adds a third field to Kamil's database of married and single people. This field is called `Years Word` and contains the strings `"ten"` and `"twenty"` corresponding to the 10s and 20s in Kamil's Years field. We ask Kamil if he could use this field to separate the married people from the single people. He says yes, but then pauses and asks: "Does the computer understand numbers?" We equivocate. Kamil makes a Venn diagram with loops labeled `years word >"fifteen"` and `years word <"fifteen"`. This puts all the icons in the first loop and none in the second. Kamil then tries using `CONTAINS` rather than greater than. Using the constraint `years word CONTAINS "y"` he is able to

get all the married people into one circle. Then he stops and says "uh-oh"—presumably realizing that the same trick will not work for the second circle because there is no letter in `"ten"` that does not also occur in `"twenty"`. We suggest using more than one letter. Kamil takes this advice and creates a successful Venn diagram with loops labeled `years word CONTAINS "twenty"` and `years word CONTAINS "ten"`.

Asked if he could apply a similar strategy to the earlier problems, Kamil says yes. We return to the animals database. Kamil adds a field called `Letter`, and enters the letter `"t"` for every dog and `"c"` for every cat. In the Tabletop window, two Venn loops with constraints `letter = "t"` and `letter="c"` produce the desired separation. Sure enough, he has solved the problem without numbers! As before, he resists the notion that there might be any meaning to his choice of the letters `"t"` and `"c"`. One researcher begins a question: "Suppose you were going to give this database to another person..." Kamil interrupts: "I know what you're going to say." He renames the field containing the animals' names from `Animal` to `Name`, and the field with the letters from `Letter` to `Animal`. Then he changes all the `"t"`s to `dog` and all the `"c"`s to `cat`.

Afterwards he says that "the words puzzled me for a few minutes." Asked if he would used numbers or words in future situations, he says he'd use whichever was easier.

Let me offer an interpretation of this session that is undoubtedly not the whole story but helps to illuminate the ideas of this paper. In the first problem, Kamil decides to try to tell the computer who is a boy and who is a girl, but he is troubled by the realization that the computer will not know the meaning of the words. So instead, he devises a method, using fictional numerical weights, that relies not on meaning but on mechanism. (By this I mean that the weight numbers are used not as representations of fact, but as devices which take advantage of known patterns in the tool's behavior.) With prompting from the interviewers, Kamil is able to apply a mechanism-oriented approach to letters as well as numbers. Finally Kamil returns to the use of words, but this time with an understanding that (in our words) words in a database can be treated mechanistically by the computer, while being meaningful to a human reader.

The same story can be retold in terms of transparency. Textual data in database programs is mediated through what are termed *character strings*. Up to this point in his interactions with Tabletop, I postulate that Kamil had used character strings transparently, seeing only the words that were spelled in the strings. The group separation task prompted Kamil to reconsider this assumption in the light of his beliefs about what a computer understands. By analogy with numbers, which he knew how to manipulate mechanistically, Kamil adopted a method based on operations like equality and substring search (called "CONTAINS" in Tabletop), in which

strings behave not as words but as character strings. The use of opaque notations like "c" and "t" initially facilitated his work at this new symbolic level. At the end, however, he was able to coordinate a use of character strings in which meaning and mechanism each have their place.

12.3 Transparency and Kindred Concepts

En route to clarifying what transparency may mean in this context, let me mention some other terminologies that cluster around this issue. "Direct manipulation" (Shneiderman, 1983) was the design goal of the 1980s. "The systems that best exemplify Direct Manipulation all give us the qualitative feeling that we are directly engaged with control of the objects—not with the programs, not with the computer, but with the semantic objects of our goals and intentions" (Hollan, Hutchins and Norman, 1986). One of the most important principles associated with this idea is that of coordinating a program's output and the user's input within a single representation, which could therefore take on continuity as a "model world" during the interaction.

For a more experience-oriented view, we can turn to Heidegger. His vocabulary of "readiness-to-hand," "breakdown," and "presence-at-hand" was explained and promoted by Winograd and Flores (1986) as a way of analyzing the phenomenology of tool use, with explicit attention to the ways that our awareness of mediating devices may vary. The essential idea is that as long as a tool is "ready-to-hand," i.e., functioning in a way that allows it to be taken for granted, it does not exist as a distinct object for the user. A hammer being used to drive a nail, for example, "is part of the hammerer's world, but is not present any more than are the tendons of the hammerer's arm....The hammer presents itself as a hammer only when there is some kind of breaking down or unreadiness-to-hand. Its 'hammerness' emerges if it breaks or slips from grasp or mars the wood, or if there is a nail to be driven and the hammer cannot be found." (Winograd and Flores, 1986) These terms could be applied to Kamil's experience: Initially ready-to-hand, Tabletop's handling of words became present-at-hand early in the session under the force of his own questioning.

Long used in semiotics, the term "transparency" has been revived within a culturally oriented perspective (Wenger, 1990; Lave and Wenger, 1991; Meira, 1991), wherein its meaning is both well-developed and ambitiously broad. Lave and Wenger (1991) characterize transparency of technology as "the way in which using artifacts and understanding their significance interact to become one learning process." The clinical examples presented here exemplify such an interaction, I believe. However, for these writers, significance can extend to the many "fields of meanings" of an artifact within a culture (Wenger, 1990). In the case of a tool like Tabletop, these might include the role of data analysis and statistics in scientific and political discourse. While such indirect meanings are indeed essential in learning to use such a tool well, the tool is not used for the purpose of representing them. Such meanings are not what I am talking about in this paper.

I am concerned here with the more basic case of a mathematical device that is being used to model a situation of interest. For such a modeling relationship, the metaphor of transparency carries a useful set of implications. It acknowledges a model's mediating role by placing it between the user and the domain of interest. It suggests how one might not be aware of the model's presence, even while depending on it. "Transparent" thus captures much the same notion as "ready-to-hand," but it seems more apt for symbols and representations. (Perhaps "ready-to-eye" would be a more Heideggerian coin.) At the same time, we know that transparent objects are not always invisible—they can be looked at as well as through (which is equivalent to the ready-to-hand/present-at-hand distinction).

The metaphor of transparency can also be misleading. It can be taken to imply that a transparent medium reveals objects as they "really" are, rather than as medium-dependent constructions. It also suggests that transparency is a property of the medium itself, independent of the perceiver's learning, whereas transparency can the hard-won outcome of learning to use a medium well. But there is a kind of transparency that we do not work to achieve, and which seems to reveal objects as they "really" are: This is naive transparency, and it happens as long as, and insofar as, our prior constructions of the domain are not challenged in our experience of the new medium. As Andy diSessa has said, "transparency is best, not when it is given, but when it is achieved" (personal communication). But let us not look down too much on naivete: It is an essential and inescapable dimension of our being. In transparency there is always a dynamic mixture of the naive and the reflective, the given and the achieved.

12.4 The Transparency Dialectic

Lave and Wenger (1991) make this important statement about transparency:

> It combines the two characteristics of *invisibility* and *visibility*: invisibility in the form of unproblematic interpretation and integration into activity, and visibility in the form of extended access to information. This is not a simple dichotomous distinction, since these two crucial characteristics are in complex interplay, their relation being one of both conflict and synergy. (p. 103)

I would like to use some slightly different language to describe what this complex interplay can be like. We may take Kamil's session as an example of a recurring phenomenological-epistemological progression in the transparency of computer tools, which might be called the "transparency dialectic." This dialectic opposes *naive transparency* (thesis), to *opacity* (antithesis), with *coordinated transparency* as a synthesis of the two poles. Naive transparency might also be called naive realism: The user does not suspect the existence of some mediating layer or component. In opacity, the mediating layer is problematic and the situation of interest is not directly visible; the tool's mechanisms must be reflected on, and solutions are based purely on mechanism, as when Kamil cautions that, "This isn't going to

make any sense." In coordinated transparency the user achieves a synthesis of meaning and mechanism: She/he can understand and accommodate the requirements of the medium's mechanisms, but is usually able to act and think as if there were no such layer in between. This is a desirable outcome. To quote Wenger (1990), "representation artifacts must become invisible for the learning to be fully integrated in some ongoing activity."[2]

It might seem needlessly confusing to use the term "transparency" for both the beginning and end states of this dialectic. Clearly they are very different kinds of experience. But "transparency" may be a good name for what's similar between them. Consider two people interacting together around a computer environment, making transparent use of some tool component. Suppose that for one person the transparency is naive, and for the other it is coordinated. As long as the use of the tool component does not become too problematic, their behaviors will be very hard to distinguish. In all likelihood the two people themselves will not be aware of a difference in their experience of the tool. This similarity is very important, with many implications for communication and learning processes.

Kamil's use of numbers as a transitional solution brings up some interesting subtleties. It seems reasonable to say that numbers served as a model of how a symbol could be used in a non-transparent way. However, the opacity of Kamil's numbers is not exactly parallel to that of his letters. The character strings that mediate the input and output of numbers, unlike those used for words, are normally transparent, because a string of digits can be treated by the computer as a true number, susceptible to all the standard arithmetic operations.[3] Kamil uses digit strings transparently: It is the numbers themselves which are opaque. Kamil knows that the computer can manipulate numbers (note that he asks "Does the computer understand numbers?" only later in the session, referring to spelled-out numbers). And yet Kamil's use of numbers is not as definitely opaque as it would have been had he simply made a field called Number, with meaningless numbers; instead, he makes fields (Calories, Pounds, Years) which are meaningful, but fictional. The role of fiction is intriguing. Perhaps the fictional structure supported his use of the tool's numerical mechanisms by keeping a clearer connection to his past uses of the tool. One might also say that through the device of fiction, Kamil turned the modeling relationship around, making a fictional world serve as a cognitive interface to the tool's abstractions.

The potential for turning around the modeling relationship is a benefit of transparency. A good modeling tool embodies mathematics in a way that makes it easier to "see" the subjects to which that mathematics is applied; once these subjects are

[2]See diSessa (1986) for a similar progression described in terms of functional and structural understanding.

[3]The digit string layer can occasionally become salient when students accidentally set the type of a numerical field to "string," leading to unexpected tool behaviors.

in view, they become a background against which the mathematics itself can be better seen. Words can illuminate the workings of character strings; weights of boys and girls can illuminate the transformation from data table to Venn diagram.

Clinical Example #2: The Transparency of Structure

Kamil's preoccupation with the meaning and meaninglessness of words was unique in our study. (Not, I would argue, because the issues are irrelevant to other children, but because it was only with him that we witnessed them being worked out.) What about the other subjects? More commonly, subjects ran into difficulty not with the words themselves, but with the way the words needed to be arranged in order for the computer to deal with them correctly. Independently of each other, most of the subjects (9 out of 13) created similar data structures of a kind that, although intelligible to humans, is quite inappropriate in the Tabletop program. They added not one but two columns to the database—one for girls and one for boys. In each column they simply listed all the names of that gender, without regard for the alignment of information in rows (a typical example is shown in Figure 9). This led, of course, to bizarre data records. Double-clicking on icons in Tabletop, for example, yielded object descriptions such as in Figure 9(B).

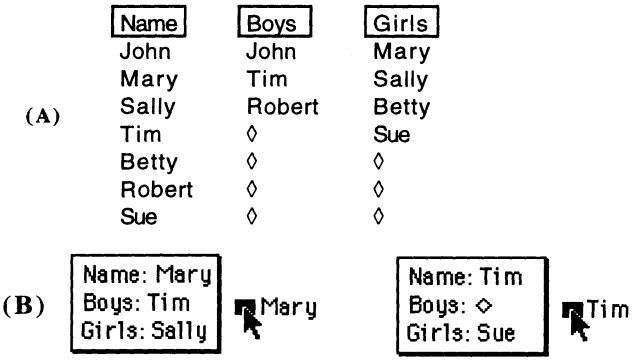

Figure 9. An unsuccessful attempt to solve the group separation problem. Inappropriately structured data in (A) produces confusing records as in (B).

Some vocabulary would be helpful here. We will call the representation consisting of a field of individual names and a parallel field of corresponding properties a *property-based* representation. In contrast, a *set-based* representation presents equivalent information in the form of sets, each with its own name and list of members (Falbel and Hancock, 1993).

Property-based

Name	Gender
John	male
Mary	female
Sally	female
Tim	male
Betty	female
Robert	male
Sue	female

Set-based

Boys	Girls
John	Mary
Tim	Sally
Robert	Betty
◊	Sue
◊	◊
◊	◊
◊	◊

Figure 10. Two fundamentally different representations of the same information.

Either kind of representation can be used to describe the same underlying logical relationships. While most database programs, including Tabletop, require an attribute-based representation for data entry, it appears that, at least in instances like this one, the set-based form comes more naturally to youngsters than does the attribute form. Many of our subjects struggled through, often with some hints, to an attribute-based solution, but would then slip back to a set-based orientation on follow-up problems. There are many interesting questions to consider regarding the conditions of and reasons for students' apparent preference for set-based representations. One conjecture is that many of the facts recorded by students in databases—ages, weights, calorie counts, etc.—may fit more naturally for children in an attribute-based format, as compared with a binary boy/girl distinction, which seems to be more naturally represented by children in a set-based format. While students had definitely made and used binary fields in their databases, it is possible that these were all modeled by a teacher, so that it was not up to the students to choose an attribute-based representation over a set-based one (Falbel and Hancock, 1993).

Here, however, my interest is in the fact that the difficulty encountered by these students represents, once again, an opaque intrusion of a domain of the tool that had, up to that point, been transparent to them. In this case the domain is the data structure on which the tool operates. Just as character strings, together with their associated operations and relations, such as substring search, can be thought of as a mathematical domain, so can the tool's underlying data structures—the records and fields visible in the data window—together with the transformations that generate possibilities for plots and graphs in Tabletop window. These transformations form a mechanistic layer of the tool, which operates independent of meaning. Of course, meaning is not irrelevant: The structures and transformations have been designed so that whenever the program rearranges the words in the database, they still make sense and reflect the truth. If, in one row of the data window, we enter `female` under the `Gender` heading, and `Rashida` under the `Name` heading, we are not surprised when later `Name: Rashida` and `Gender: female` appear together when a particular icon is double-clicked. Because both these arrangements of words are meaningful to us, we do not have to think about them in terms of underlying mechanisms. That is, the mechanistic layer can be transparent, as long as we have matched our meanings to the structure in accordance with the tool's design. From the point of view of these

students, even with many hours of Tabletop experience, the data entries in the data window were simply reasonable ways to describe the facts, and the computer's representations of the facts were unproblematic.

In terms of the transparency dialectic, for many of our subjects the group separation task presented difficulties that disrupted the naive transparency in their experience of data structure in the software. With the exception of one subject who developed a coherent explanation of the problems of set-based representations, most subjects who eventually solved the problem muddled through without a clear sense of why one method worked and another didn't. These subjects still have a distance to go in exploring the opaque mechanisms of data structure. The experience of working on this problem may at least have helped to raise the issue for them.

What was it about the group separation task that precipitated the breakdown of transparency? The group separation problem begins with a goal expressed in terms of arrangements of icons, and asks what prior condition is necessary in the data window to achieve it. As already noted, this is the reverse of the transformation that the software makes automatically, and it is also the reverse of what the students had generally been asked to do.[4] In their previous projects the students had first created databases, designed either by adults or by themselves. Then they had explored in the Tabletop window to see what graphs they could make. With little prior expectations of what graphs they would be making, they had no reason to notice if a particular kind of graph was possible or not possible. The motto "Aimlessness perpetuates naive transparency" may capture some of the problem here. An exploratory, come-what-may orientation may have been appropriate for a while, but I think these students would now benefit from more planful, goal-oriented activities in which the affordances of the tool would be subject to more critical scrutiny.

12.5 The Tool and the Curriculum

The question of how a teacher might manage the vicissitudes of transparency raises important issues of curricular and pedagogical judgement. By solving some problems ahead of time, and steering activities around potential difficulties, teachers can (and routinely do) allow aspects of a tool to remain invisible in students' experience. Conversely, teachers can make it more likely that an invisible transparent layer will come into view, and can support groups or individuals in reflecting on it and arriving at a coordinated understanding of it. The judgement about how to pace this process, i.e., the process of the transparency dialectic, can be based on all kinds of considerations, including classroom goals and how they are being served along the way, as well as students' readiness for frustration and complications. The judge-

[4]As Andee Rubin points out (personal communication), this feature of reversal is shared with other difficult tasks, such as medical diagnosis and the construction of mathematical proofs. We know a solution when we see it—but how to create one?

ment can be better made to the extent that the teacher is aware of that layer and the circumstances under which it is more or less likely to intrude into the class's awareness.

Though timing may be under the teacher's influence, the fact remains that, no matter how supposedly "transparent" the tool design, without a teacher's vigilance to prevent it, any part of the mathematical model that is being looked through will eventually need to be looked at. Rather than struggle against this powerful trend, I would rather adopt a view of computer-based tools, and the mathematics they embody, that harmonizes with it. Let us think of a tool not only as a device to make useful things happen, but also as a way of packaging some mathematics that we judge to be worthy of students' attention. Let us consider the transparency dialectic itself to be a process which students might have reason to practice and get better at. In short, let us adopt the principle that, to paraphrase McLuhan, the medium *is* the curriculum. This principle has several implications.

First, tool designers should acknowledge that they are designing not just a tool but a curriculum. The implicit mathematics of a tool needs to be brought into the open and weighed in the same way that any kind of curriculum is weighed: for its accessibility and interest to children, its cultural significance, its contribution to intellectual and social empowerment. At times, general-purpose computer environments have been portrayed as a radical alternative to the oppression of planned curriculum. While the rigidity and materialism of overly scripted, instructionist curricula is certainly something to be overcome, adults should still make careful choices about the artifacts, ideas and activities with which we ask children to engage. Indeed, Papert's *Mindstorms* (1980), which some have read as a manifesto against curriculum, is an eloquent curricular argument, and appropriately so. When, for example, he portrays the computer as an entity that "speaks mathematics," or argues for the general utility of skills such as debugging and making subprocedures, Papert is making an argument for the curricular value of LOGO and similar environments based in part on the mathematics they embody.

Second, the implicit mathematics of a tool needs to be judged on its form as well as its content. Is the implicit mathematics rendered in a coherent and developmentally appropriate fashion? One example of a problem of coherence comes from a commercially available educational data analysis program that accepts data in columns (Figure 11). Among the graphing options possible with this tool, one is a box plot by groups. Here the implicit mathematical model is that each column represents a group of cases, with the numbers representing some measurement on members of the group. However, another option will render the same numbers as a scatter plot. Now the implicit mathematics is different. Both columns now apply to the same group of cases, with each column representing a different attribute. (And suddenly unequal column size is a problem, where before it wasn't.) To use the terminology introduced earlier, this tool wavers between a set-based and an attribute-based interpretation of rows and columns. Almost never would both interpretations make sense for the same input. This is not good tool design. While the tool provides

a useful set of capabilities, the underlying mathematics is incoherent. When issues of data structure become problematic, the tool's mixed messages are more likely to lead to confusion than to enlightenment.

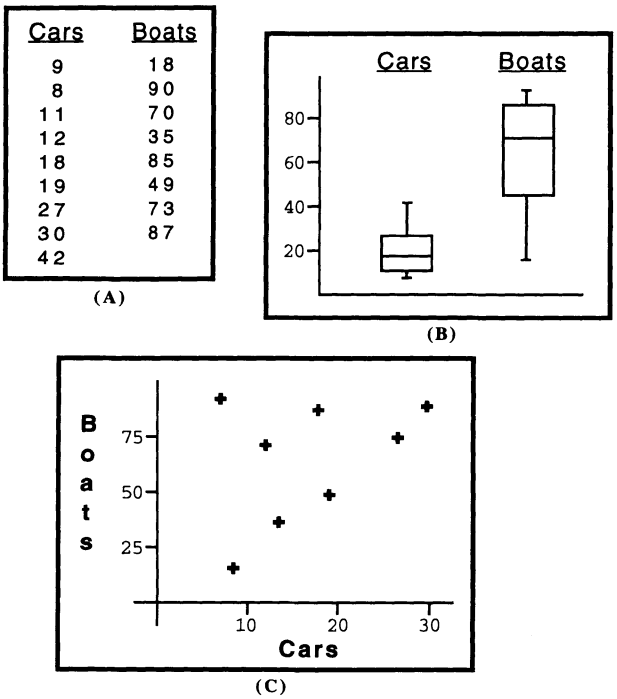

Figure 11. An example of incoherence in a mathematical tool. Plots (B) and (C) represent conflicting interpretations of the data structure in (A).

Another aspect of mathematical coherence is completeness. For example, Tabletop has a shortcoming that is shared by all current educational data tools: It is incomplete with respect to the transformation of data structures. Most obviously, the reverse transformation from set-based to attribute-based representations should be possible. This would allow children to enter data in a set-based form when it better matched their understanding of the situation. But more generally, the domain of data structure should be rendered as a transformational space, which students could explore in order to make sense of structures and transformations as a coherent system. The point of such a data tool would not be to prevent the mechanisms of data structure from ever intruding on students' experience; the relationship between set-based and attribute-based representations, for example, is still important to learn about. But the intrusion could happen a little later, a little more gently, with the help of more cognitive support and the reward of greater empowerment within the environment.

Incidentally, I can describe this ideal tool in terms of its general properties, but exactly how it should behave is not obvious. Mathematics and computer science offer a hodge-podge of data structure concepts, more or less conditioned by historical limitations and idiosyncrasies of computing technology: table, relation, array, matrix, variable, list, vector, tuple, tree, pointer, queue, stack, record, object, and scores more. A coherent, accessible theory of data structures and transformations would be a significant design accomplishment—and its acceptance would be a major political accomplishment! This brings me to my third point.

The third implication of "the medium is the curriculum" is that, where the mathematics of a computer tool is judged worthwhile, it needs to be integrated into the accepted body of mathematics curriculum. This may entail some reorganization of topics and branches of mathematics. However, a first step is simply recognizing the existence and importance of the mathematical ideas that are embodied in a tool.

The mathematics of data structures is a prime example. Data structures and their transformations deserve first class citizenship in the mathematics of the 1990s. At present data structures are taught in the U.S. only at the college level, as second-year computer science. But data structures are not just the province of programmers. They are absolutely fundamental to data analysis as well as to the use of most computer-based tools. Data structures and their transformations are also very powerful mathematics, with strong connections to other kinds of mathematics like algebra, linear algebra, symbolic logic and other discrete math topics, and, of course, computer programming. One way to integrate data structures into the mathematics curriculum would be to incorporate them into discrete mathematics—balancing what I consider to be an overemphasis on algorithms in current visions of the field—and also to build a connection back from college level discrete mathematics all the way to attribute blocks and sorting activities in the first grade. Data analysis and data manipulation environments would be an important part of that connection. In addition, algebra curriculum could be revised to build on the conceptual similarity between data "fields" and algebraic variables. (By suggesting new ways to think about mathematical topics, I don't mean to imply that mathematical pedagogy should be heavily topic-oriented. The best learning about data structure, and indeed all of these topics, may happen in interdisciplinary contexts.)

Before such changes can be contemplated, the discourse of educators, mathematicians and statisticians will need to acknowledge the ubiquity and fundamental nature of data structure concepts. These concepts have a peculiar kind of invisibility now. Adults who must use data structure concepts every day, in doing statistics, say, or using various computer tools, do not seem to be aware of concepts they are using, or of the need to pass them on to novices. Polanyi (1958) described a category of knowledge which he called *tacit*. Among other things, tacit knowledge is indispensable in making sense of seemingly straightforward linguistic communication. Likewise, data structures are a kind of *tacit mathematics*, indispensable to the

intelligibility of data analysis and data tools, straightforward though these may seem to expert users.[5]

The advent and maturation of computer-based tools and their use in classrooms provides an opportune time for bringing tacit mathematics to light. Tools are part of the trend away from learning *about* mathematics and science, towards learning to *do* mathematics and science. As this transition progresses, children in their engagement with tools help to expose the gaps between theory and practice. It is an interesting twist on the phenomenon of transparency, where our own knowledge can itself be invisibly transparent, so that we don't realize what knowledge we draw upon in order to use a tool or technique successfully. Fortunately, our own transparency dialectic is underway. Reflection on newly recognized transparent knowledge can help mathematicians and educators in reconstructing a mathematics in which more students will find clarity.

References

diSessa, A. (1986) Models of computation, in D.A. Norman and S.W. Draper (eds.) *User Centered System Design: New Perspectives on Human-Computer Interaction*, Hillsdale, NJ: Erlbaum

Falbel, A. and Hancock, C. (1993) Coordinating sets and properties when representing data: The group placement problem, *Proceedings, Seventeenth International Conference on the Psychology of Mathematics Education*

Hutchins, E.L., Hollan, J.D., and Norman, D.A. (1986) Direct manipulation interfaces, in D.A. Norman and S.W. Draper (eds.) *User Centered System Design: New Perspectives on Human-Computer Interaction*, Hillsdale, NJ: Erlbaum

Lave, J. and Wenger, E. (1991) *Situated Learning: Legitimate Peripheral Participation*, Cambridge: Cambridge University Press

Meira, L. (1991) On the transparency of material displays: The case of learning linear functions, Paper presented at the Conference of the American Educational Research Association, Chicago, IL

Papert, S. (1980) *Mindstorms*, New York: Basic Books

Polanyi, M. (1958) *Personal Knowledge: Towards a Post-Critical Philosophy*, Chicago: Univ. of Chicago Press

Shneiderman, B. (1983) Direct manipulation: A step beyond programming languages, *IEEE Computer*, 16/8, 57-69

Wenger, E. (1990) Toward a theory of cultural transparency: Elements of a social discourse of the visible and the invisible, Doctoral dissertation, Dept. of Information and Computer Science, UC Irvine

Winograd, T. and Flores, F. (1986) *Understanding Computers and Cognition*, Norwood, NJ: Ablex

[5]Readers alert to the messages hidden within the normally transparent medium of English orthography will note that the letters of the word "mathematically," when rearranged, spell "mayhem, all tacit."