CHAPTER 6

More Ideas to Help You Build Better Environments

Purpose of This Chapter

The previous two chapters set forth the fundamental method for constructing Environments. This chapter is an elaboration of the method along with responses to criticisms of the Environment idea.

Review of What an Environment Is

A complete Environment is a database whose *content* is a mathematical subject, or, indeed, any technical subject, and whose *access* has been optimized for speed of problem solving. Normally, databases contain facts about the real world, e.g., in a bank, the financial records of each customer, or, in a manufacturing firm, employee records, or, in a lab, scientific data. But there is no reason why a database can not contain a technical subject, e.g., linear algebra.

Databases are accessed by queries. The user issues a command such as, "Give me a list of all the employees who are under 30 years of age and are earning more than $30,000 a year." In the case of a computer implementation of an Environment, typical queries might be, "Give me a definition of the term *eigenvalue*" or "Give me a list of all the theorems that contain the term *orthonormal*".

But an Environment is not merely a database. (Any textbook can be regarded as a database.) *An Environment is a database optimized for rapid problem-solving in a technical subject*. Keep this in mind if you find yourself getting involved in long-winded discussions and arguments about Environments.

Unfortunately, neither complete computer database Environments nor complete word-processor Environments nor complete printed Environments exist yet. (The author is working on the first one, for linear algebra.) So, for the time being, you have to build your own (partial) Environments. This book tells you how to do this using long-hand or a word-processor.

Before we proceed, however, it is important that you understand what an Environment *is not*.

What an Environment Is Not

An Environment is *not*:

- A collection of programs, like Mathematica, Mathlab, StatView and numerous others, for solving certain types of mathematical problem.
- An online representation of a mathematical text, with hypertext, computer-generated indexes, and a search facility.
- A collection of cookbook procedures which, if you follow them exactly, will enable you to solve a (limited) class of problems without your knowing, or being able to quickly find out, why each procedure works.

Straight Talk on Hypertext

Hypertext is the electronic linking of marked words and phrases in online text so that, when the reader clicks one of these marked words or phrases, a related text is immediately displayed. Hypertext is a simple, obvious, inevitable electronic implementation of an index. Yet careers have been built, families raised, kids sent to school, on this word, which is very impressive in the high-tech world. (Anything with "hyper" in it is impressive in the high-tech world.) Believe it or not, there is even something called "hypertext theory" (ye gods!) which is nothing more than the slow, painful, re-invention, by dim, liberal arts minds, of a few of the programming practices that were already well-established in the early seventies.

The point is, you should not be deluded by the argument occasionally raised against the Environment concept that "it is all just a matter of hypertext", or, for that matter, "all just a matter of search facilities", the latter referring to those programs that find, in a text, all occurrences of the words or phrases you specify.

An Environment is a structure aimed at problem-solving efficiency and, at least at present, and for the foreseeable future, Environments can only be constructed with human intelligence, specifically, human intelligence that understands the structural principles described in the chapters, "Fundamental Concepts" and "How to Build an Environment".

Why the Environment Concept Is Important — And Inevitable

Even if the old way of learning mathematics — namely, by learning the logical structure of each subject — were determined to be the one and only best way to acquire depth of knowledge of a subject, *nevertheless the old way is doomed*, for the simple reason that there is far too much mathematics to attempt to master in that way.

"...we have amassed more knowledge since World War II than all the knowledge amassed in our 2-million-year evolution on this planet. In fact, the amount of knowledge that our scientists gain doubles approximately every 10 to 20 years." — Kaku, Michio, *Hyperspace*, Anchor Books, N.Y., 1994, p. 274.

"During the past fifty years, more mathematics has been created than in all previous ages put together. There are more than 1,500 mathematical research journals, publishing some 25,000 articles every year (in over a hundred languages). In 1868 The *Jahrbuch ueber die Fortschritte der Mathematik* (Yearbook of Mathematical Research) listed just twelve categories of mathematical activity; in *Mathematical Reviews* for 1992 there are more than sixty." — Stewart, Ian, *The Problems of Mathematics*, Oxford University Press, New York, 1992, p. 19.

I doubt if any mathematician would deny that the amount of mathematical knowledge has become so great that no one person can have the remotest hope of mastering it all, or even of knowing what, in the way of concepts, not to mention lemmas and theorems, is out there.

Specialization is thus a necessity, and it has been made into a virtue. I have known mathematicians who clearly felt that you really had no reason to be walking the face of the earth unless you knew as much about their specialty as they did. On the other hand, I have known recent PhDs in mathematics who decided not to pursue academic careers because, after all the labor and expense and time of getting a PhD, they were appalled at the vast areas of mathematics that they still knew nothing about.

This situation might not, in itself, be enough to call into question the old way of learning mathematical subjects *if* there were a guarantee that all problems in any given specialty could be solved solely within that speciality, and similarly for sub-specialties, and sub-sub-specialties, etc. (An interesting question! What kinds of logical structure have this property?) Unfortunately, there is abundant reason to believe that this is not the case.

"In the 1870s mathematics consisted of a number of separate and diverse disciplines, such as arithmetic, algebra, geometry, and infinitesimal calculus. In the 1970s [and, presumably, later] this is no longer the case. There has been an enormous increase in the volume of material which a mathematician has to master, but there are no longer any boundaries between the different 'branches' of the subject, which are now so closely interdependent that no mathematician can afford to specialize. In the

1970s [and, presumably, later] to be a good mathematician is to be a polymath." — Temple, George, *100 Years of Mathematics*, Springer-Verlag, New York, p. 282.

It is simply not possible to master all that material in the old, classroom way of studying all the theorems, and their proofs, and working all the exercises so that we get a B or an A on any closed-book exam in the subject. It is no longer humanly possible, just as it is no longer humanly possible to do the world's calculations by hiring people with the gift of doing rapid calculations in their head.

Expressing it in the language of computer scientists who specialize in database technology: the old way *does not scale*. As the amount of mathematics grows that may be required for the solution of a given problem or class of problems, the number of human beings capable of solving the problem shrinks rapidly — fewer and fewer persons have the ability to memorize and integrate such a volume of material — and the time required to solve the problems grows ever greater. *This is not a desirable state of affairs!* (No matter how flattering it may be to the exceptional few.)

(A well-known mathematician has remarked that the explosion of mathematical knowledge suggests that there are too many mathematicians! To me, this is an astounding indication of how stuck in the old ways of thinking some professional mathematicians are when it comes to the manner in which mathematics is presented (hence taught and learned). It is as though a well-known computer scientist, looking at the growing size and number of computer programs in the sixties, had announced that it is clear there are too many programmers!)

To anyone who understands the Environment concept, and in particular structured proof, it is clear that the exploding amount of mathematical knowledge and of the size of some proofs does not mean that mathematics is approaching some kind of limit, any more than the increasing size and number of programs means that computer science is. But mathematics has not yet had its Edsger Dijkstra, who already in the sixties recognized that a prime goal of programming theory must be what he called "the control of complexity".

We — the vast majority of those who study and use mathematics — must give up the old idea of mastering each subject, and instead settle for the new idea of "just-in-time learning": being able to move rapidly from subject to subject, solving the most common problems in each with a minimum of intellectual labor and with the help of paper and electronic tools (Environments) designed to aid us in this purpose.

But then, sooner or later, an inevitable question arises: **Why do we study**? The traditional answer is: so that we will have the knowledge to solve problems. We learn the material so that we can use it. But, as I hope is clear by now, we can use a subject (in the form of an Environment) without having learned it! We can look up what we

need to know to solve a given problem. *Studying, in the traditional sense, is becoming obsolete*.

If you are still skeptical that the old ways of presenting mathematical knowledge need to be improved (or can be improved!) imagine that this country was engaged in a war in which victory would go to the side that could demonstrate the greater speed at creating and understanding difficult proofs. Do you think we would continue in the way we have for the past 2300 years?

Advantages of Environments

Environments Give Students *and* Professors What They Want Most

Ask students what they most want in any mathematics course, and the answer will most often be, "Formulas, short-cuts, ways to get the right answers!"

Ask professors what they most want in their careers and the answer will most often be, "Not to have to teach, so that I can do research instead![1]"

Environments are the best means I know of to give students *and* professors what they want. In fact, this was one of the original motivations for developing the Environment concept.

Environments Make All Math Subjects Look "the Same"

A good description of Hell is: waking up each morning knowing that everything you learned yesterday is different today — or rather, knowing that *some unpredictable subset* of the things you learned yesterday will be different today, in an unpredictable way — as, e.g., in the computer industry. Once you have learned something, it is easier to learn something that is almost the same than it is to learn something that is different. Obvious, you say. Yet this obvious idea seems to have escaped teachers of mathematics and authors of mathematics textbooks. "The "Universal Template for Mathematical Entities"" on page 73, and, indeed, Environments in general, are an attempt to make all mathematical *concepts* similar to each other *from a problem-solving point of view*, just as structured proof is an attempt to make all *proofs* similar to each other *from a structural point of view.* Of course, you may point out that the traditional logical format in which almost all mathematics courses are now presented — definitions, theorem, proof, theorem, proof, ... — that this format already makes all

1. Well, no professor will say this out loud. You have to read studies of the academic mathematics culture to discover this. See, e.g., Morris Kline's *Why the Professor Can't Teach*.

mathematics subjects similar.The trouble is, this format shows us the *logical* similarity of mathematical subjects. But that is by no means the only kind of useful similarity, as I think Environments will demonstrate to you. *Problem-solving similarity* is equally useful. If every subject looks "the same" from a problem-solving point of view, then, when you come back to the subject after having been away from it for a while — over a summer vacation, or after one or more semesters — it will be much easier to resume problem-solving in it.

The desire to determine the similarities between things is a common one in mathematics. In the forties and fifties, a subject was invented (or discovered) which shined a new light on the similarity of mathematical subjects. This subject is called "category theory". Very roughly, it views every subject as consisting of a set of objects and a set of functions relating the objects to each other. Then, applying this whole idea "one step up", it proceeds to consider subjects themselves as objects, and to relate them to each other via another set of functions. This simple idea has been of great use in research in many branches mathematics.

They Enable You to Solve Problems More Efficiently

When I say that Environments result in greater problem-solving efficiency I mean the following: randomly divide all the students taking a given math or other technical course into two groups. (I assume the course is taught by only one professor.) Let one group study and take exams as they normally would. Let the second group study and take exams using a *complete* Environment for the course. Have both groups keep track of all the time they spend on the course. When the final grades have been published, assign 4 to an A, 3 to a B, 2 to a C, 1 to a D, and 0 to an F. Let each student in each group divide the numerical equivalent of his or her grade by the total number of hours he or she spent on the course. The resulting number is what I am defining as problem-solving efficiency. I claim that the average efficiency will be higher for the second group than for the first.

Students and Others Can Learn New Subjects Faster

I continue to be amazed that this book wasn't written by someone else ten or twenty or fifty years ago, because as long ago as that it must have dawned on many people that the explosion of mathematical knowledge which has occurred in the twentieth century carries with it a new problem, namely, the problem of how to get that knowledge by all those who need it when they need it. Yet courses still last a quarter or a semester — weeks, months! — and a bachelor's degree still requires four years.

It is often said that you need to go to school in order to learn *basics*, the implication being that, once you have learned these, you will be able to learn advanced subjects on your own if necessary. Pompous educators in speeches before entering freshman and before graduating seniors often talk about school being a place where you "learn how to learn". But never once — not once! — in many years of sitting in classrooms did I ever hear a teacher say anything remotely like, "And, by the way, when you need to learn something knew about a subject like this, some useful tricks are ..." The truth is that not one teacher in a thousand knows beans about self-teaching because the education of those in the teaching establishment makes abundantly clear that such knowledge, and the dissemination and discussion of such knowledge, is distinctly *scientia non grata*. (If you teach people how to learn things themselves, well, then, soon there will be no need of Us!)

I sometimes wonder how the typical math professor imagines his *best* students will go about acquiring new knowledge in his subject after they have left school. The idea that every time someone in industry needs to apply a branch of mathematics or any technical subject which they are not familiar with, they should enroll at a nearby university and take a course (if it is being given) surely must have struck *even* a few *professors* as outrageously inefficient.

Mathematics and indeed all technical subjects are used — applied — by many persons who are not experts. Think of all the people who use probability and statistics in their jobs. Think of all the engineers who use calculus, the programmers who use compiler theory. The truth is that most users of mathematical subjects learn as much as they need and little more. The truth is, there is less and less time to learn new subjects via the old process of studying (possibly in a course), memorizing, and practice-applying over a period of several months.

In any case, what makes subjects easier to learn and apply for people outside of school, may also be of help to people in school, namely, to students.

The existence of Environments for each technical subject, I am convinced, will significantly decrease the time required to earn a degree in a technical subject. In an age when technical knowledge is growing at the rate it currently is, that is *not nothing*.

They Enable You to Get Back Into a Subject Faster

Every student has had the experience of suddenly needing to be able to solve problems in a subject he or she has partially forgotten, and therefore having to scramble through old notes and textbooks in order to recapture the necessary knowledge. Environments reduce this effort significantly.

Less Need to Memorize

Why do we need to memorize so much of what we study? Because it is the fastest way to access the knowledge we need when we need it. We memorize so that it is "all there" when we need it. We memorize because, up to now, our memory has been the best rapid-access database available for holding the knowledge we need to solve problems. (Of course, we also memorize because that is what our teachers believe is a major part of learning.)

But even so, in the modern world we memorize *much less* than the ancients, say, in Greek and Roman times did. I believe that, if any thinker or scholar or poet or dramatist from those times were brought back to this world with roughly the same faculites he had when he left it, we would be astounded at how much he had committed to memory. Long texts, entire poems — pages and pages and pages.

Yet I think most would agree that the choice our civilization made was the wise one, namely, to trade the instant recall of memory for the much vaster memory space represented by writing and then, later, by printing, and now, of course, by computer memoies, just as I think you will agree that our civilization made the right choice in developing calculating machines instead of trying to find and train more and more people with extraordinary abilities at doing large calculations in their head.

On the other hand, repeated use of knowledge or information — as in generating and using Environments — usually means gradual memorization, so that repeated use means faster problem-solving. Which is precisely how problem-solving speed should come about: not because smart students have memorized more theorems and formulas, but because experienced students need to spend less time looking up this knowledge in the first place. (In a complete Environment, every student can look up the same knowledge equally fast.)

You Can Sell Copies of Your Environments

Under "Assume You Will Forget!" on page 93, I said that one way of to create a good Environment is by imagining you are creating it for another student. If you do your job well, then you may be able to sell copies of your Environments to other students. Since all Environments have the same structure, your customers can then augment and improve your Environment from where you left off, without having to start from scratch. This will doubtless be easier if your Environments are in the form of word-processor files, but pencil-and-paper Environments can also be improved by the next user.

The Environment Concept Does Wonders for Your Self-Confidence

In the last analysis, this may be the major benefit of Environments. You will find that no matter how intimidating, how unimaginably difficult a new technical subject appears to you, you will be able to say to yourself, "I may not know anything about the content of this subject, but I know *how it goes* from a problem-solving point of view." You will know what shape you need to put it in for maximum problem-solving efficiency. You will know that there is nothing profound about the author's or the teacher's need to present the subject to you in its logical structure, or about the absence of pictures and guides to intuition, or about the need for you to memorize everything in order to accomplish anything. You will have a healthy contempt for being forced to do more intellectual labor than is necessary in this knowledge-crowded world, and you will have the tools to start reducing this labor.

It Is Easy to Test How Well Environments Work

It is tempting for me to say that Environments work if you, the student, think they do, e.g., if you think they enabled you to get a higher grade. But the best argument for (or against) the worth of the Environment concept is a statistical one. Such an argument can be easily made by the kind of test described under "How the Environment Idea Should Be Tested" on page 130. As you can see, unlike most educational theories, the worth of the Environment concept is easy to test.

Disadvantages of Environments

It would be irresponsible of me not to set forth some of the disadvantages of Environments. Here are the main ones I have discovered. See also "Why Go Through the Trouble of Building an Environment?" on page 82.

Alphabetical Organization Seems Odd to Some Students

The most frequent criticisms I have received of the alphabetical organization are "It's too hard to read," "It doesn't flow," this despite my careful explanation in the first-time user information referenced on the Start Page that, unlike the typical textbook, this one you don't have to read first before you try to solve problems; that you simply look up things when you need them, that this is just-in-time learning.

The attempt to organize mathematical knowledge alphabetically has given me a first-class lesson in the difficulty which most people have in confronting what is known as a *paradigm shift*, meaning a major change in the way they think about a

given subject or task. One mathematician remarked, "What do you think mathematics is, some kind of alphabet soup?" A number of people have even commented that headings such as "integral, multiple, how to compute", "curve, determine tangent at a point" don't look right because they contain commas (!). This is how deeply oriented to syntax many people in technical fields are.

Difficult Problems Require Extensive Page-Turning

In pencil-and-paper Environments, a lot of page-turning may be necessary, particularly in the case of definitions which use terms that require further look-ups, and in the case of proofs with several sub-levels. When things seem likely to get out of hand, you can use pencils or paper clips to mark where you have been.

The equivalent of page-turning in on-line implementations — e.g., using hypertext — goes much faster.

Building an Environment is More Tedious than Taking Notes

Let's see how much more tedious the building of an Environment is than taking notes. If you are building a pencil-and-paper Environment, then, yes, as you take notes, you have to decide where the notes go — under which topic, and you have to decide if you want to cross-reference them from other topics. You can — *and should* — take lecture notes directly in Environment form, i.e., put them under the appropriate alphabetical heading and cross-reference as you go.

On the other hand, you will not have to do a linear search through your notes every time you want to look up something.

Certainly, converting typical prose proofs into structured proofs is more time-consuming. (See the chapter, "Proofs".) On the other hand, you save time when you need to review those proofs.

Adding diagrams to word-processor Environments is often very tedious unless you have a good graphics program. These are becoming available.

Pencil-and-Paper Environments Require Lots of Paper

Partial Environments for some subjects, e.g., calculus, linear algebra, may require two 3-inch loose-leaf binders, assuming hand-written pages and one page per topic. Creating paper implementations on the computer vastly reduces the amount of paper required because a topic can always be inserted between two other topics (to maintain alphabetical order) without necessarily producing a new page in the printout of the Environment.

You Need to Keep Track of Logical Order of Theorems

You need to keep track of the logical order of lemmas and theorems. Otherwise you risk committing the logical fallacy of begging the question, that is, of assuming the truth of something which you are trying to prove. This problem is avoided in traditional textbooks because *logical* order is what is typically made clear by the order of presentation. A simple way to avoid the problem in Environments is to use the numbering of theorems and lemmas given in the main textbook you are working from, and then checking carefully before you use a higher numbered theorem or lemma to prove a lower numbered one. If you are working from several textbooks, and, possibly, lecture notes, you will have to interpolate the appropriate numbering.

The problem of logical order in Environments has never been a serious one in my experience.

You Will Become Impatient With Traditional Textbooks and Courses

Once you start thinking about technical subjects in terms of Environments, you may become impatient with traditional textbooks and classroom teaching. In class, you will sometimes feel like you are being forced to listen to someone read the t dictionary to you. I'm afraid I can't offer any remedy at this time.

Criticisms of the Environment Concept

Most of the criticisms of Environments I have heard so far fall into the following categories. Responses are given immediately following each criticism.

1. **Environments are just cookbooks**, and no student (other than a student cook) ever learned a subject by following cookbook procedures. In order to learn a subject as difficult as most mathematical subjects, there is no replacement for hard work and drill. You simply have to struggle until you've mastered it, meaning, until you have understood it. As Euclid is reputed to have said to the king of Egypt, "There is no royal road to learning."

Reply:

I never heard of a student who got all or most of the problems right, being accused of not understanding the subject.

The term cookbook normally means that the user is not given an explanation of why the steps are as they are. But that is emphatically not the case with an Environment. An Environment is structured so that it is much easier to find out answers to questions than in a traditonal textbook, or in student notes.

Learning is a red herring. So is understanding. Yes, I know how outrageous those two statements sound to most ears. What the statements mean, of course, is that making learning and understanding the primary focus, diverts our attention from the business at hand, which is the ability to solve problems. Learning and understanding *should be* side-effects of creating, using, and improving Environments. The student has a perfect right to say (fully realizing it is an exaggeration): "I don't want to learn. I want to *do*!"[1]

Furthermore, I am certainly not proposing that the traditional form of textbook be eliminated! It is perfectly legitimate for a student, or anyone else, to want to see the logical construction of a subject. Both Environments and traditional textbooks can, and will, exist side by side.

If you can find theorems and proofs rapidly, this will encourage you to look them up. Furthermore, you will have a new way to simplify and understand proofs, as described in the chapter, "Proofs".

Finally, I must add that almost invariably when I ask math teachers and professors what they believe students most want in a math course, the reply is, "Formulas and cookbook procedures for solving problems! All they want to do is to be able to solve the homework and exam problems as quickly as possible." This is meant as *criticism* of the students! I, on the other hand, think that a teacher or a professor has an obligation to deliver to his or her customers the product that they want! And the truth is, these customers, namely, the students, live in a culture in which solving problems — whether in the SATs, or in classroom examinations, or in Graduate Record Exams, or on the job — is really all that counts. I think that the argument that the "student must understand" is as often as not a smokescreen to hide the ineptitude of classroom teaching and of textbook authors.

It is an invoking of the logic of psychotherapists: if the patient improves, that proves how valuable psychotherapy is; if the patient doesn't improve, or grows even worse, that is because he didn't try hard enough (or, in classroom terms, didn't attempt to "understand" the subject matter). I say again: learning and understanding should be *side-effects* of creating, using, and improving Environments.

---

1. The sentence is a paraphrase of one used in an ad on some of the classical music stations in the San Francisco Bay Area in the early 2000s, the ad being for a company (whose name I have forgotten) that offered to do Internet searches. In the ad, an exasperasted computer user says to her primitive search engine: "I don't want to search. I want to *find*!"

2. **Environments discourage memorization**.
Reply:
On the contrary, they are an aid to memorization because they enable you to quickly look up what you need to memorize, and because they put information into a form that is designed for rapid understanding hence (I argue) rapid memorization. The rigorous structuring of proofs as described in the chapter, "Proofs", is certainly an aid to memorizing proofs.

3. **Environments discourage thinking.**
Reply:
The only kind of "thinking" they discourage is struggling to recall material that may or may not have been learned before. If you agree that a person with a photographic memory still needs to think — in other words, a person who can recall anything he or she has previous read or learned — then you agree that the use of Environments does not discourage thinking.

How (and Why) the Environment Idea Was Developed

Several readers of this book have asked how the Environment was developed. The main impetus to its development was my going back to school in my thirties to get a Master's degree in Computer Science. I had had only two-and-a-half years of electrical engineering education some fifteen years previous and here I was planning to do a Master's thesis on one of the most abstract and difficult new developments in computer science, namely, Scott and Strachey's Mathematical Semantics of Programming Languages, a theory which made possible the first rigorous definition of the semantics of programming languages. Very few computer science professors, and none at the university where I was working on the Master's, understood the theory, which had only been published a few years before. So I had to teach myself. I found that grinding the material out of books and academic papers in the traditional way was simply too difficult. I had to find a better way.

But there were earlier motivations. As an undergraduate, I had always been forced to learn according to someone else's idea of how learning should be carried out. In particular I regarded with contempt the whole idea of prerequisites for a course, because it seemed to me silly in its naivete. There it was, late in September, and the course catalog would list, say, Calculus 101 and 102, Analytic Geometry 75, and

perhaps one or two other courses as prerequisites for, say, Calculus 110. Did the plodding academic fathers seriously imagine that students *remembered* the contents of all those courses, some of which they might have taken a year or two previously? If not, then just exactly how much did the fathers imagine, or hope, or demand, that students did remember, and where did they spell this out? The fond academic fantasy of *good* students reviewing *all* their courses throughout the summer, going over their notes at every spare moment, nauseated me. Only a college professor could be capable of such delusions.

As I began developing my first Environments, I began getting a different view of the task of solving problems in a course. Instead of looking at it as an activity designed to separate the winners from the losers I began asking myself what it would be like if the purpose was to grind out as many correct answers as possible. I imagined I was the Secretary of Education during a war in which, for some reason, victory depended on the problem-solving *productivity* of students. The more correct solutions to problems that were produced, the more likely would be our victory. How would I organize textbooks and courses if that were the case? Certainly not by the traditional method of students taking notes at lectures, then cramming for exams, a process which, like prerequisites, seemed silly, because after you had passed the exams and the course, what did you have? Rapidly fading memories of knowledge you could only retrieve with difficulty. Of course, the teachers and professors always had an out: they could always say, "Well, God knows I taught them, I covered all the material in the syllabus, I tested them. If they can't remember it, well, that's not my problem." What a racket!

Of course, I was worried from the start about the *weakness* that the need for Environments revealed. I thought: let's face it, an Environment is a prosthesis, a device for the mathematically handicapped, something we use to get around in (get around a subject in), like a wheelchair or artificial leg. But then I thought: so what? so is a car or an airplane or a calculator. If an Environment enables me to accomplish intellectual tasks I couldn't accomplish otherwise, well, hooray for Environments!

I began wondering about how the Environment idea could be tested

How the Environment Idea Should Be Tested

As more and more students start using Environments, inevitably there will be arguments and discussions about whether Environments are any good or not. One way of deciding is simply by polling students and determining if their math grades increased after they started building and using Environments. But that would not be a

good argument for the value of Environments, since other factors may have been at work. A more rigorous way of deciding — and the way that I hope will eventually be used — is by a statistically significant set of comparison tests such as the following.

1. At the start of any university class in mathematics or any other technical subject, divide the class, by random selection, into two groups.

2. Now with neither group having received any classroom instruction or homework, give the midterm exam to both groups. One group must use the standard textbook and other course materials, the other a *complete* Environment for the subject. Each student is to work alone and may take, say, two weeks to answer as many questions as he or she can. (The time period may be shortened to only several hours, with the students required to take the test in the classroom. In this case, there might be a loss of statistical validity due to fewer problems being answered.)

3. Compare scores and times for completion for the two groups.

My claim is that the Environment group will perform significantly better. In fact, I suspect that several students in the textbook group will be able to answer few, if any, questions.

A third kind of test is what we might call The Turing Test for Environments, after a test developed in the early fifties by the mathematician Alan Turing. That test was aimed at determining if a computer was intelligent or not by comparing the computer's responses to questions to a human's responses, with both sets of responses being communicated via computer terminals. In the Turing Test for Environments, we ask if an objective observer can tell the difference between a student who had completed the course and one who hadn't taken it but who had an Environment at his or her disposal, the questions being taken from exams given throughout the course, but with the *time that each student took to answer the questions* being *removed* from the answers given to an objective judge.

Environments and Artificial Intelligence (AI)

"Artificial intelligence" (AI) is a term applied to computer behavior when this behavior is considered as approximating what we regard as intelligent behavior in humans. Examples are certain kinds of symbolic reasoning, e.g., the proving of theorems, the diagnosing of diseases from a list of symptoms, and the ability to react to commands expressed in natural, as opposed to computer, language.

In the mid-eighties there was a brief flurry of interest in AI in the computer industry (as opposed to merely in academic research laboratories). This period was sometimes referred to as "AI Spring" by researchers who felt that at last the time had

come when some of their efforts would find commercial applications. Unfortunately, it was soon followed by "AI Winter".

At the same time, computers were being applied to aid in the design of integrated circuits, specifically, to the problem, called the "placement and routing problem", of arranging transistors on a chip so that they, and the metal traces connecting them, would take up the least space on the chip, thus allowing the physical size of the chip to be as small as possible.

At the company where I was working, programmers tried to make the programs do these tasks without human intervention. It was an all-or-nothing proposition, no doubt because programmers' egos would be much more satisfied if their creations could do the entire job unaided. What brilliance that would demonstrate!

I argued for interactive programs in which the user could say, in effect, "OK, try a placement in which the following transistors... must be placed as indicated..., and these others... as indicated..., but the rest can be placed whereever you want, and then show me the results..." Then the user could manually move transistors and/or wires and tell the program to try to find a placement with the placement of these transistors and/or wires being fixed. I pointed out that the interactive approach *contained* the non-interactive one, in that one could simply allow the program to run to completion without subsequent user intervention. Nevertheless, I lost the argument. It is not quite correct to say that modern placement and routing programs are not interactive. It is just that the interactivity takes place at the program design level, not at the chip design level, by which I mean that, nowadays, the programs themselves are constantly modified in an attempt to improve their performance. But once modified, they run to completion, or until they run out of memory space, or time.

It is a shame that a few tests weren't run of the present way of doing things vs. the interactive way, because I believe that there are occasions when human *visual* intelligence is capable of seeing, in a few seconds, solutions to problems which would take a machine many minutes if not hours to arrive at.

Similarly, the Environment concept will someday yield computer programs designed to be used interactively. Students (or other users) will be able to tell the computer, "Spend five minutes trying to find the following integral...", "Spend ten minutes or so trying to prove the following..., starting at ..., and using the following theorems..." Then, depending on the results, they will be able to repeat the command, perhaps suggesting other strategies.

I argue that a student should only have to demonstrate *once* that he or she has discovered a useful heuristic for solving a class of problems. This heuristic may, of course, be embedded in a computer program. Thereafter, the student should merely have to set up the problem — e.g., a triple integral — and then ask the computer to do

the rest, perhaps aiding it by given suggestions, e.g., specifying the order of integration.

Another example is: "Graph this function..." (Have you, as a student, *ever* been asked to write down a useful procedure for performing this task that works in the vast majority of cases?)

Along the same lines is the following thought.

Euler-on-a-Chip

Leonhard Euler (1707-1783) is generally considered one of the four greatest mathematicians who ever lived, the other three being Archimedes, Newton, and Gauss. He was certainly the most prolific among them. His total output runs to some seventy volumes and has not been completely published to this day. Yet to me — and I must emphasize that, first, this is a personal opinion which many mathematicians will probably disagree with, and, second, that this is the age of the computer — many of Euler's discoveries seem to be the result not so much of conceptual breakthroughs as of an extraordinary facility with algebraic manipulation. By this I mean that these discoveries might have been made by a human, far less talented than Euler, but working with a computer, perhaps even a present-day computer, which allowed the user to enter the equivalent of commands like, "Beginning with the following equation ..., see if you can come up with an equation of the form ... Quit if you haven't found anything after 24 hours." An example of this kind of problems is:

"Beginning with the following series,

$$\sum_{n=1}^{\infty} \frac{1}{n^s} = \frac{1}{1^s} + \frac{1}{2^s} + \frac{1}{3^s} + \dots$$

see if you can come up with an equation of the form,

$$\sum_{n=1}^{\infty} \frac{1}{n^s} = \Pi(\ldots)$$

where "..." contains only primes p and the positive integer s."

The answer, or rather, an answer, is Euler's famous product formula,

$$\Pi\left(\frac{1}{1-\frac{1}{(p^s)}}\right)$$

where the product is taken over all primes p.

I point this out because I think it is important for students to recognize — to develop an intuition about — the difference between algebraic manipulations (no matter how complex or ingenious) and new ideas, e.g., the idea of a group, the idea behind Cantor's proof that there are at least two different infinities. In the age of the computer, it is natural to become impatient with problems which are merely manipulations.

Of course, there is a gray area here when we talk about proving theorems. Some proofs are almost a matter of filling in the blanks, given other theorems, others are not.

My point is that you should not hate yourself when you sense that the mathematical labors before you are not intrinsically difficult, but merely tedious. It is important to recognize this.

Undergraduate-on-a-Chip

Let me be frank: one of my goals is to create a set of Environments that will enable a motivated high school graduate to solve all the problems in a typical undergraduate math curriculum — or in fact in any typical curriculum in the hard sciences or engineering — just as fast as he or she can. That includes doing proofs and answering essay questions. I simply don't see the benefit of forcing students to sit through x number of hours of class. Time-serving is for prisoners.

"It is a profoundly erroneous truism, repeated by all copy books and by eminent people when they are making speeches, that we should cultivate the habit of thinking of what we are doing. The precise opposite is the case. Civilization advances by extending the number of important operations which we can perform without thinking about them." — Whitehead, Alfred North, quoted in Newman, James R., *The World of Mathematics*, Vol. 1, New York: Simon and Schuster, 1956, p 442.

Maybe years from now, for high school graduation gifts, some parents may give their kids something called, say,"Undergraduate-on-a-chip", which will be a set of Environments convering all of undergraduate technical subjects, or least math subjects. These Environments will certainly have access to programs like Mathematica, and, possibly, to "reasoning modules" like that described under "Euler-on-a-Chip" on page 133.

Notation is Not Merely Symbols

At some point in the mathematics education of most students, they hear about the importance of "notation" in mathematics. The typical example used is the Arabic number system — meaning our familiar base 10 number system, which originally came from the Hindus — and the second typical example is the algebraic notation used to solve word problems, e.g., equations like

$$x^2 + 3x - 4 = 0$$

The third typical example, if the school offers a course in calculus, is likely to be the *dx*/*dy* notation of calculus.

"[Edsger Dijkstra:] It's very illuminating to think about the fact that some — at most four hundred years ago — professors at European universities would tell the brilliant students that if they were very diligent, it was not impossible to learn how to do long division.

"[Interviewer:] Long division?

"[Dijkstra:] You see, the poor guys had to do it in Roman numerals. Now here you see in a nutshell what a difference there is in a good and bad notation. A hard university subject can be brought to such manageable proportions that it can be taught in primary schools. You have exactly the same thing with Greek mathematics, which also used letters to denote digits, and therefore didn't have the notation necessary for

the introduction of variables. Add the fact that they gave all their proofs not in a formalization, but in natural language, and you see that some of their proofs doubled, or tripled, and how certain parts of mathematics were a closed book to them.

"[Interviewer:] All because the tools were wrong.

"[Dijkstra:]Yes.

"[Interviewer:]And tools make the difference.

"[Dijkstra:] They make a tremendous difference. Because tools have such a profound influence on the thinking habits of the people who are trying to use them." — Cashman, Michael W., "An Interview With Professor Dr. Edsger W. Dijkstra", in *Datamation*, May 1977.

From Dijkstra's words, and the above examples, it is not surprising if the student concludes that "notation is symbols". But that is not necessarily so.

A good notation is one that makes an intellectual task easier — in our case, a problem-solving task, or, more vaguely, a task of "understanding" the concepts of a technical subject. It is true that a set of symbols is one way of doing this, but another way is to *organize information* in a way that makes it much easier to use. Thus, a table — whether it be a table of logarithms or a table of integrals or a table of prime numbers — is a kind of notation in this sense. So is a dictionary or an encyclopedia. And so is the organization of technical knowledge which an Environment is. *Structure is notation! Structure can do some of your thinking for you!*

A way to keep yourself on the structural track as you create Environments is to follow, as closely as you can, the rule: *Get rid of prose!* Let structure (and pictures and horizontal curly braces — see "Get Rid of the Equals Sign!" on page 111) *show* what the prose in textbooks usually *says*, just as Viete's new algebraic notation in the late 1500's *showed* what pages of prose previously tried to *say.*

In my experience, computer scientists seem to have a much more robust and creative attitude toward notation than mathematicians do. The reason is probably that computer scientists have been forced, by the very nature of hardware and software, to confront the issue of What vs. How, semantics vs. syntax. Every mathematics student should take a few computer science courses that stress the importance of the difference between semantics and syntax. (A course in the programming language LISP is certainly among these.) As one of our best writers on mathematics has put it:

"...the symbolism of mathematics is merely its coded form, not its substance." — Stewart, Ian, *The Problems of Mathematics*, 2nd ed., Oxford University Press, N.Y., 1992, p. 9.

"Mathematics is not about symbols and calculations. These are just tools of the trade — quavers and crotchets and five-finger exercises. Mathematics is about *ideas*.

In particular, it is about the way that different ideas relate to each other. If certain information is known, what else must necessarily follow?" — ibid., p. 10

"Someone once stated a theorem about prime numbers, claiming that it could never be proved because there was no good notation for primes. Carl Friedrich Gauss proved it from a standing start in five minutes, saying (somewhat sourly) 'what he needs is *notions*, not *notations*'." — ibid., p. 10.

What Should You Be "Allowed" to Look Up?

The better an Environment, the less work you have to do to find what you need in order to solve problems. But sooner or later, you will begin to wonder how much you should be "allowed" to look up in the course of solving a problem. In the case of the problem in "A Trigonometry Problem" on page 219, should you be "allowed" to merely look up theorems on bisectors of angles and find the one that solves the problem? Shouldn't you concede failure because you didn't remember that particular theorem (if you ever learned it at all)?

The answer to these questions is simple: suppose you had a photographic memory and you never forgot anything. Would you allow yourself to use anything you had memorized? I suspect your answer is yes. Therefore: you can use anything in any Environment (or in the extended Environment which is the textbooks you have available to you) — anything that helps solve the problem, *provided* it does not beg the question, i.e., provided it does not assume what you are trying to prove, and, of course, provided the problem statement itself does not explicilty prohibit it (as in cases where you are asked to prove something without using a certain theorem).

Another answer is this: suppose online versions of standard math textbooks are made available to students. Do you think you should be allowed to use the standard search facility which will no doubt be available with these online books to search for theorems containing a given word or phrase?

"I began to read the paper [on nerve impulses in cats]. It kept talking about extensors and flexors, the gastrocnemius muscle, and so on. This and that muscle were named, but I hadn't the foggiest idea of where they were located in relation to the nerves or to the cat. So I went to the librarian in the biology section and asked her if she could find me a map of the cat.

"'A *map* of the *cat*, sir?', she asked, horrified. 'You mean a *zoological chart*!' From then on there were rumors about some dumb biology graduate student who was looking for a 'map of the cat.'

When it came time for me to give my talk on the subject, I started off by drawing an outline of the cat and began to name the various muscles.

"The other students in the class interrupt me: 'We *know* all that!'

"'Oh,' I say, 'you *do*? Then no *wonder* I can catch up with you so fast after you've had four years of biology.' They had wasted all their time memorizing stuff like that, when it could be looked up in fifteen minutes." — Feynman, Richard P., *Surely You're Joking, Mr. Feynman!*, W. W. Norton & Company, N.Y., 1985, p. 72.

"The important truth which seems to have been understood, implicitly at least, from the very beginning [of Greek philosophy] is that learning is not a process of dishing out information. Some of this, of course, there must be. But it is neither the sole function of the teacher, nor yet the most important one. This is indeed more evident today than it was at that time, for written records were rarer and harder to find then than they are now. With us, it stands to reason that anyone who can read can collect information from a library. Less than ever before should a teacher need to pass on mere information...The role of the teacher is one of guidance, of bringing the pupil to see for himself." — Russell, Bertrand, *Wisdom of the West*, Crescent Books, Inc., London, 1977, p. 68.

So, to answer the question that is the title of this sub-section: You should be allowed to look up whatever is look-up-able. The rule for authors of textbooks and of complete Environments is: whatever *can* be made look-up-able, *should* be made look-up-able.

Types of Computer Implementations of Environments

There are at least four types of computer implementations of Environments. These are listed below. The first two you can implement yourself. All you need is a word-processor with the usual text editing facilities, plus math fonts, sufficient graphic capabilities to allow you to do simple, labelled, drawings, and a text search facility. Either of the second two types of implementations will have to be produced by a commercial software publisher (or an entrepreneur, perhaps yourself!) since they involve extensive programming.

- an off-line textual implementation in which the text is simply entered via a word-processor and stored in text files for printing out whenever desired. Such an implementation requires a word-processor with extensive mathematical symbols. Highly desirable, but not necessary, is some form of macro capability which will enable you to define the formats of certain types of text, e.g., theorem statements, and

then have the word processor automatically put the text (when it is suitably indicated) into that format.

- an on-line textual implementation intended for use *on* the computer itself. Such an implementation is enormously simplified if the word-processor allows hypertext.
- an on-line database implementation. Here you actually input as much of the subject as you wish into a database, then access it to solve problems. Thus, e.g., you might ask for all theorems which contain certain words (or synonyms thereof).
- an on-line database implementation with graphics, animation, and low-level reasoning facilities. Here you will be able to ask the computer to graph an equation involving two or three variables, or various projections of functions involving more than three variables. Animation will be available where appropriate to aid in the explanation of concepts. Low-level reasoning facilities will allow you to enter requests such as "Spend up to about three minutes seeing if you can prove this lemma... Try using the following theorems..."

"Environmental Consciousness"

You will find that as you create Environments you will begin developing "Environmental consciousness", meaning that you will begin developing the habit of observing (and writing down!) what is difficult for you in each subject and what you find could have been understood more rapidly had it been presented differently. Probably the three most common questions that experienced users of Environments ask are:

1. "Why is this difficult?"
2. "How do I wish this had been presented?" A variant of this question is, "How could this have been presented so that I could have understood it 'at a glance'?"
3. "What do I spend most of my time on in the course of solving problems in this subject?"

Importance of a "Start" Environment

Once you have developed several Environments, you will begin to notice a need for an Environment which serves a similar function relative to all your other Environments as the Start Page serves relative to a particular Environment. Call it a *Start Environment*. It primarily contains *references* to other Environments, but it also

may contain information in itself, especially on topics which are found in many different subjects, or on subjects for which you have not yet had to start building an entire Environment in itself. Thus, as an example of the first case, you might put all the theorems and lore (and references to same) which you come across concerning Pascal's triangle, in your Start Environment. Whenever you find, in the course of problem-solving, that you don't know in which Environment a theorem or lemma or concept explanation which you need, is located, you add the correct reference to the Start Environment after you have found it.

More on Heuristics

The invention (or discovery) of mathematical proof by the ancient Greeks was unquestionably one of mankind's greatest achievements. The detailed history of how the invention occurred will probably never be known, but conceivably it began with the question, "How do we know that this statement is true in *all cases*?", e.g., "How do we know that the only triangle that can be inscribed in a semi-circle — *any semi-circle* — is a right triangle?" (The answer, "Because it always has been true in the past," was a perfectly good answer for the practical applications that mathematics was put to in pre-Greek cultures.)

Yet the question, "How did you go about discovering this proof?", or, more generally, "How did you go about solving this problem?", receives almost no attention in the university curriculum. I think part of the reason is that mathematics professors, like the professors of all scholarly disciplines, have felt it important to maintain their priestly mystique and to exclude those not deemed eligible for admittance to the inner circle. The temptation must be particularly strong in mathematics, the one discipline which is, in fact, in possession of eternal truths and a rigorous means of demonstrating them (the dream of every priesthood that ever existed!).

"...mathematical results are published and taught quite openly, but there is very little explicit teaching on how to do mathematics, and publishing besides the results also the heuristics that led to them is regarded by many as 'unscientific' and therefore, bad style: quite often the editor's censorship will try to prohibit publication." — Dijkstra, Edsger W., "Craftsman or Scientist?", address to conference on computing, 1975, photocopy, p. 4.

"As for publications, mathematicians nowadays are almost forced to conceal the way they obtain their results. Evariste Galois, the young French genius who died at the age of 21, in his last letter before his fatal duel, stressed how the real process of discovery is different from what finally appears in print as the process of proof. It is

important to repeat this again and again." — Ulam, S. M., *Adventures of a Mathematician*, Charles Scribners Sons, N.Y., 1976, p. 271.

"...Polya's larger work on the same subject [heuristics], *Mathematics and Plausible Reasoning*, has been cooly received by the mathematical community and has had at most a very minor influence on the teaching of mathematics at university level. Its cool reception by the mathematical community says at second thought, however, nothing against the feasibility of Polya's project. On the contrary! For its cool reception by the mathematical community can also be interpreted as the rejection by the mathematical guild that feels threatened, as all guilds do, when the secrets of their trade are made public. To publish 30 years ago a book about the making of mathematical discoveries was heresy, as it still is in the eyes of many mathematicians today." — Dijkstra, op cit., p. 5.

Rightly or wrongly, I can't help seeing a parallel between learning mathematical problem-solving in the West, or at least in the U.S., and learning to read in pre-modern China. In both cases, the task is (was) accomplished only by a very small percentage of the population, and only with great difficulty except for the tiny handful for whom it comes (came) "naturally". Please understand that I am *not* suggesting, via the following quotation, that mathematical notation is analogous to Chinese script. What I *am* suggesting is that the (ancient) way we teach mathematics is analogous to Chinese script, and that the Environment concept is analogous to Western writing systems.

"...in China, while you will find great numbers of people who know the significance of certain frequent and familiar characters, you discover only a few whose knowledge is sufficiently extensive to grasp the meaning of a newspaper paragraph, and still fewer who can read any subtlety of intention or fine shades of meaning. In a lesser degree this is true also of Japan. No doubt European readers, especially of such word-rich unsystematic languages as English or Russian, vary greatly among themselves in regard to the extent of books they can understand and how far they understand them; their power varies according to their vocabularies; but the corresponding levels of understanding among the Chinese represent a far greater expenditure of time and labour upon their attainment. A mandarin's education in China was, mainly, leaning to read." — Wells, H. G., *The Outline of History*, Doubleday & Company, Garden City, N.Y., 1971, p. 492.

The main reason I wrote this book was to try to convince you, the math student (and, possibly, your teachers), that the old idea of "learning" this or that portion of a math subject so that *then* you can solve problems in that portion of the subject, is an extraordinarily inefficient way of approaching mathematics; that a far more efficient approach is to *start with the classes of problems to be solved*, and proceed from there.

This approach shines a light, for the first time, on the problem-solving *attempt* itself; it puts in its proper place the traditional obsession with speed which has always been part of mathematics courses; it encourages you (and, I hope, your teachers) to *study* your own problem-solving attempts, to observe what you spend most of your time on and to think about ways to reduce that time, to control the thrashing which is so often the only real heuristic that math students apply, to start thinking about the problem-solving *craft*, to free yourself of the old obsession with answers, the old obsession with either-or, winner-loser, and instead to start thinking about good problem-solving craft vs. bad problem-solving craft.

As far as I know, this approach — at least as implemented in this book — is new. Among other things, it makes clear that books on problem-solving invariably *assume* (unconsciously) the existence of an Environment, i.e., one in the student's head. How many times have you read, in these books on problem-solving, statements like, "At first this looks like a very difficult problem, but if we just think it through and remember the rule, 'Keep it simple!', we soon see that..."; or "By working backwards, it soon becomes clear that we can apply the law of cosines..."; or "By following the rule, 'Try to convert the problem into a problem you know how to solve', we see that we can apply the transformation ... to each region and ..."

"A Trigonometry Problem" on page 219 will, I hope, make clear the difference between such superficial heuristics and those which are an inherent part of a good Environment.

There was a saying in computer science circles in the seventies, "You don't understand something until you can program it." I say, "You don't understand how to solve a given class of problem in mathematics until you are able to write down a procedure for at least *approaching* any problem in the class — for at least making an attempt that would get you more than, say, 65% partial credit in any exam." Not all the problems in the class of problems may be solvable! For example, not all functions are integrable, and, of those that are, the techniques required to perform the integration may be unknown at any given stage of a student's mathematical development. Or they may be too difficult for students at a given level in their mathematical education. Nevertheless there is a workmanlike approach to doing integration. I think it is a scandal that beginning calculus students are not encouraged to develop, *and write down*, such an approach, or failing that, that such an approach is not given to them. The argument that they must discover it for themselves is hyprocritical if the teacher does not encourage them to write it down and then use it and improve it through the doing of problems. Otherwise, the argument is just more teacherly empty talk. Similar remarks apply to the solving of differential equations[1]. On the other hand, if a derivative of a function exists, then there is always a way to find it — an algorithm

exists to find it. Similarly, although there is no algorithm for finding proofs, *nevertheless* there is a workmanlike way to go about searching for a proof, as explained in chapter, "Proofs".

Let me go further and state boldly: *calculus is the kind of thing that belongs in a machine*. Professors should make every effort to prevent students from confusing concepts and calculations. Students should be encouraged not to "demonstrate knowledge of", or to demonstrate memorization of formulas and rules but instead to demonstrate the effectiveness of written-out procedures for solving various classes of problems. The answer to the question, "Do you know how to do derivatives and integrals?" should be, "Here are the procedures that I wrote to perform these tasks." The equivalent of some of these procedures may, of course, have been implemented in commercially-available programs.

Render unto the machine, that which is the machine's, and unto humans, all the rest

In fact, let me go further and assert, *A calculus textbook is a great big calculator*. I use the term "calculator" here for emphasis: it would be more correct to say "is — or should be — a great big computer program.". As you apply the Environment idea to calculus, you will understand more and more why I say this.

What we want — what we should be striving for — is a conscientiousness about heuristics — about making them explicit — that matches our conscientiousness about logic.

I will conclude this section with a second answer to the question, "aren't heuristics at the center of mathematics teaching?" The second answer is that heuristics seem a rather peripheral concern to teachers who never themselves have to learn new, back-breakingly difficult subjects. Primary, middle, and secondary school teachers teach the same math subjects year after year. Some may attempt to increase their knowledge of these subjects, and/or improve their skill at teaching them. But these are entirely different matters from attempting to learn how to solve problems in new subjects.

---

1. As far as I know, to this day there is not a published table or flow-chart that allows a person wishing to solve a differential equation to start at the top and work his way down to a list of the names of methods that can be used to solve that particular equation. Instead, the student is expected to take sufficiently many courses so that he can create the equivalent of such a table in his mind. Yet another example of the shocking backwardness and professor-serving nature of the mathematics culture.

Essay Questions on a Math Exam? Yes!

One of the iron-bound articles of faith of mathematics teaching is that the only acceptable demonstration of understanding is the ability to solve homework and exam problems. The ideal class of students, for many college math teachers, would be one which could be given theorem, proof; theorem, proof; ... *and nothing more*, because that is, after all, the essence of any mathematics subject. (As elsewhere in this book, I am assuming that theorems include the statement of formulas for solving problems). By studying the theorems and the proofs, this ideal class would then be able to do the homework problems and get good grades on exams. They would "understand" the subject.

These same teachers, however, if asked to evaluate a student's (or a prospective colleague's) understanding of a subject in a face-to-face interview, do *not* ask the interviewee for a recitation of theorems and proofs, nor do they give the interviewee a random selection of textbook problems to solve. They instead ask for what amounts to a restatement of some of the key ideas of the subject in the interviewee's own words (including, of course, the key ideas of some of the most important theorems). They probe how well the interviewee knows the "lay of the land" of the subject. This metaphor is entirely appropriate, because the teachers are proceeding as though "understanding the subject" meant having a big diagram (map) of the subject somewhere (in the interviewee's mind and/or on paper and/or — who knows? — on the walls of the interviewee's study) and then being able to talk informally about the contents of the diagram, just as one who has drawn or otherwise obtained a map of a country and acquainted him- or herself with it, is able to talk informally about the geography of the country with an ease that is the envy of listeners who don't possess that map.

The late John Holt, unlike virtually all other experts in education, actually spent time in the classroom, and not only that, spent time talking to students (in this case, primary school students) trying to find out how they *thought about* what they supposedly knew.

He was amazed to find that some students who got consistenty high math grades often had very little conceptual grasp of the mathematics which, on the basis of their performance on homework, classroom and exam problems, they supposedly "understood" . Instead, they had become extremely skillful at reading the teacher's non-verbal clues as to the right answer.

One way of building conceptual understanding is by talking and writing about concepts, specifically, by responding to questions like: "Describe how you go about solving a problem of the type ..." (i.e., give your heuristic for solving this type of problem); "Draw a map (e.g., a Venn diagram) of all the types of" (such-and-such

entity) "which we have studied in this subject;" "Describe as clearly as you can the parts of this subject which you find most difficult and try to give reasons why you find them difficult;" "State informally, in your own words, what you understand the gist of the" (such-and-such) "Theorem to be;" "What are the kinds of acceleration we have met with so far? Show which is a sub-set of which."; "If the angular speed of a ball whirled on a string is constant, how can it be undergoing acceleration toward the center of the circle?" "Describe how you might test Newton's Third Law." Essays on math exams, despite the contempt with which they are viewed by many in the mathematical community, are now becoming more common, as part of the so-called "New New Math" (which is distinguished from the "Old New Math" that fluorished in the sixties and was aimed at teaching young students set theory, arithmetic in other than the decimal number base, and other abstruse subjects).

If you have doubts about this, I invite you to look at the heuristic (here, an algorithm) under "linear congruence, how to solve a" on page 211. Given such a heuristic, and assuming that you have proved that it is correct, what is the point of working exercises that ask you to solve linear congruences (other than, perhaps, the pleasure of seeing the heuristic in action)? You have already done the important work, the hard work, in developing the heuristic.

Similarly — and I know this will sound outrageous to many math professors — I don't see the point of working calculus problems until the heuristic is in place. It is well known that an algorithm exists for obtaining derivatives, but not for doing integration. Nevertheless, there are better, as opposed to worse, ways for approaching integration problems, and your business, as a success-oriented student, is to write down a heuristic as soon as possible, modifying it as you go. Given *any* integration problem that you are likely to encounter in the present course, or that you encountered in any past course, how should you proceed? Certainly an early step will consist in checking if the integrand is a sum, in which case you will break it down into a sum of integrals.

Once you have come to agree with the heuristic-first point of view, you may may understand why I say that *calculus is the kind of thing that belongs in a machine*. I don't believe there is any great value in proving, over and over again, to a succession of teaching assistants and professors, that you can do certain basic algebraic operations which follow from the definitions of derivative and integral, even though you still make mistakes in doing them. Learn them once, write them down, and then let a computer program (like Mathematica) do them from then on. Render unto the machine that which is the machine's, and unto humans, all the rest.

On "Understanding"

A frequent criticism of the Environment idea has been that it is merely a "cookbook" approach to learning, and that students do not benefit from cookbook approaches because these approaches aren't based on "understanding". But I have never heard of a professor writing on a student's exam paper, "It is true that you got all the answers right, and that the methods you used were all correct, but since I am suspicious that you do not really understand what you did, I am giving you an F."

As I think you will agree when you have begun developing and using Environments, the method is by no means a cookbook approach in the sense that it encourages you to merely memorize formulas and procedures. All that an Environment does is *allow you* (not force you) to know *as little* as you really need to know in order to solve a given class of problem. (In the computer industry, this is called "just-in-time learning".) Its very characteristic of rapid access to the information you want, encourages you, I think, more than any traditional textbook, to investigate *why* certain procedures and heuristics work. Rapid access to what you want to know is an aid, not a detriment, to understanding.

But let us not throw the baby out with the bath! A cookbook approach is not necessarily a bad thing. The astronauts use such an approach, in the form of a check-list, whenever it is important that no step be overlooked. Scientific experiments are often conducted in the same way, in order that the researchers can be sure that variations in procedure do not influence the outcomes. You can be sure that the Viking Lander experiments in 1976 that tested for signs of life on Mars followed a cookbook approach.

You might reply that, in order to set up the the list of steps, understanding is required. But even here, I see nothing wrong with attempting to develop a checklist for all the issues that must be dealt with in an experiment that falls into a given class of experiments, e.g., issues such as (in biology) temperature control, purity of sample and of surrounding conditions, etc., and (in physics) issues such as accounting for all potential sources of energy gain and energy loss.

A cookbook approach is only bad when the user does not know and cannot find out, easily and quickly, the reason for each step.

On Drawings

(This section expands on the section, "Use Drawings!" on page 89.)

I am convinced that the vector calculus and, in particular the divergence, gradient, and curl functions, particularly as they are used in field theories in physics, could be made much more rapidly understandable by the right pictures. But no one wants to spend the time it would take to create the right pictures. Far better — far *easier* — to cover the page with explanatory prose and equations.

"The reason Dick's [Feynman's] physics was so hard for ordinary people to grasp was that he did not use equations. The usual way theoretical physics was done since the time of Newton was to begin by writing down some equations and then to work hard calculating solutions of the equations. This was the way Hans [Bethe] and Oppy [Robert Oppenheimer] and Julian Schwinger did physics. Dick just wrote down the solutions out of his head without ever writing down the equations. He had a physical picture of the way things happen, and the picture gave him the solutions directly with a minimum of calculation. It was no wonder that people who had spent their lives solving equations were baffled by him. Their minds were analytical; his was pictorial." — Dyson, Freeman, *Disturbing the Universe*, Harper & Row, Publishers, N.Y., 1981, p. 56.

"Felix Klein [one of the leading mathematicians of the nineteenth century] did not hesitate to admit, 'To follow a geometrical argument purely logically without having the figure on which the argument bears constantly before me is for me impossible.'" — Kline, Morris, *Why Johnny Can't Add*, Vintage Books, N.Y., 1973, p. 57.

"Quantitatively, a person can visually absorb information equivalent to millions of characters a second, while the normal rate for reading text is less than 100 characters per second." — Ingalls, Daniel, *Byte*, Aug. 1981, p. 168.