TEX in Schools: Just Say No

Konrad Neuwirth
Postfach 646, A-1100 Wien, Austria (Europe)
EARN/Bitnet: a4422dae@awiuni11

Abstract

TEX is a very good tool for typesetting. But does it offer anything for schools? The author explains in detail why he thinks that TEX should not go into the schools.

Introduction

After some years of fooling around with TEX and using TEX professionally, the author has started to think about the broader use of TEX. Due to his current affiliation with the Austrian school system (the author is still a student), his main focus in this paper will be his thoughts about using TEX in schools, especially in computer science didactics. This paper was written with the Austrian schools in mind and the state of computer science introduction here. Although Austria is not the only country undergoing the described problems, the author is not trying to set up any globally valid rules.

Current Situation

Before we start to discuss where TEX could come into schools, we might want to consider what should be the goals of the basic introductory courses to computers (and this is all the schools can offer to all of the students). Currently, two main approaches are taken.

Approach One: use the computer only in a special subject. Here, we still have to distinguish between the approaches:

1) Teaching the basic ideas behind applications with didactically meaningful examples or
2) teaching the basic principles of programming.

If the computer is only used in one subject (which will probably be called something like "computer science"), I find the first approach the more desirable one. The students should see the computer as a tool rather than as a toy for some freaks who just sit in their back room and hack up new programs. Here, I see no space for fitting in TEX, which I take as a programming language that pretends to be an application.

The second approach, although widely practiced, is the best way to scare people. I think having knowledge about only one programming language (which could even be Basic for that matter) is worse than knowing nothing about computers. If the first thing a user ever sees are some cryptic symbols which help with almost no work but only create more work, it is a very traumatic experience. It is hard work to convince such a user of the fact that computers can help with his or her standard work. What we will be faced with in this case won't be computer illiteracy but computer hatred. I should point out that TEX could be used in such a situation, but I suggest this is not what a teacher should do.

Approach Two: Integrate the computer into the "classic" subjects. This means that the language course teacher teaches how to use word processors, on-line spelling-checkers and thesauruses (a triple that will revolutionize language courses); math teachers will teach the use of spreadsheets and geometry teachers will teach what CAD systems can do for technical drawing.

This approach is probably the best as it makes students use computers for their real papers in the language courses and not for some texts which are just typed for the sake of learning how to word-process them in a computer course. This approach can, but need not, be combined with an introduction to computing which should start at least one or two years after the students start using computers. They first have to think of the computer as a tool like pen and paper before they can be confronted with the inner workings of computers or programming.

TEX in Computer Science Courses

As noted earlier, I think that TEX can only be presented as a programming language and not as an application program. So, in my mind, this rules out a few things already. But can we use it in the other approaches? The answer is a simple "no". TEX is

simply not a very didactic tool. There are better programming languages to show basic concepts such as recursion. There is even a language developed for teaching children: LOGO. It enables the teacher to use recursion, or all the other major concepts like functions or procedures, in a very intuitive manner. In the appendix, you will find two programs which accomplish exactly the same thing, one in LOGO and the other one in TEX. It is needless to say which is more useful in a classroom situation.

In the TEX world we have a tool which could be used in that context and which has not yet earned a lot of respect for it's potential didactic use: Literate Programming. I know of one university lecture where the lecture notes are available as `WEB` files to the students, but there is no such thing in schools. And, for me, the important thing behind Literate Programming is that it allows the reader to watch the program designer originate ideas and develop them into working programs (if the programmer does use `WEB` accordingly, of course). As unreadable as Knuth's books might be, his programs in `WEB` are a pleasure to read. The didactic value of *TEX: The Program* must not be underestimated. It is one of the best things given to a reasonably experienced Pascal programmer. And with `Spidery WEB`, we have the tools to use Literate Programming with other languages, too.

The difficulty in programming TEX to do as you want it to also creates another problem: teacher education. Although this should definitely not be one of the prime criterion in selecting educational tools, it still is a problem. To really program TEX is difficult enough. But to be good enough to further teach how to program in TEX is even harder. So we see that TEX is not really a good tool to teach computer science.

TEX in "Classic" Subjects

Maybe we can find a way to use TEX as an application? There are two more places we could try to let TEX sneak into schools: language courses and arts courses.

Language courses. Teachers of language courses teach the students how to represent their ideas on paper, although their main goal still should be to teach them how to verbally express ideas. For this, TEX is overkill. Students should be taught the basic principles of document design, but the emphasis still should be on the content, not on the presentation. If we let them use TEX, they would spend a lot of time on visual presentation and not on the content. This is because of the perfectionism TEX provokes. We already can produce very, very good output with TEX so we also have to clean up the minor glitches and re-edit the input file at least two or three times until the paper looks almost perfect. This takes a lot of time. A researcher might do that, because nobody minds if his paper is finished half an hour later, but in school, half an hour is a lot of time, especially with the tight schedules the schools usually have (in Austria, the units of subjects taught are 50 minutes). So in the time the student spends TEXing and previewing his paper, he might instead be polishing up its content.

In schools, word processors which offer some basic functions to polish up the image of a text are sufficient, even if the lines are not broken as perfectly as if done by TEX. Especially for younger students, it is not so important to have a perfect representation of what the output will finally look like on paper, but it is important that the visual representation can be changed interactively. Even tools like Microsoft Word or WordPerfect are too much because their large number of functions are not really necessary for educational use.

The basic idea behind didactic software should always be: how can I easily show what is important? If we use WordPerfect, students spend most of their time memorizing key combinations. If we use TEX, they would have to use an editor (which takes time to learn) and still remember all the cryptic tokens that TEX uses. A lot of things can be said about TEX, but not that it is either didactic or intuitive. Small word processors like MicroStar from the Borland Turbo Pascal Editor Toolbox are sufficient. (Well, not quite. There are some things it lacks too.) There are pull-down menus, so there is no need to remember the control-alt-cokebottle combinations, and all the functions are easily accessible.

Art courses. Art education does not consider typography and typesetting worthy topics of art education because there are more important things to learn. In my career as a student, I never heard how a good-looking document is produced. I learned how to paint a surreal city; I learned how to see colors, I learned about the ideas of different schools of painters. But I did not learn how to make a document strikingly different, be it due to its very special font or markup. Apart from that, the art teachers I know have some knowledge about calligraphy, which is considered an art-form, but typography is a trade.

Other courses. I see no place where TEX could be used sensibly. For instance, with current math curriculums, I don't know any place where TEX could be used. Neither can it be used as a tool nor an example to illustrate a specific mathematical concept. It might be shown as an example where different "classical" subjects can be integrated (mathematics, computer science and arts). However, this is a threatening idea to current school systems, because then they could not keep up with the very strict separation of the different subjects and would have to go to a more integrated and overall different way of teaching. Although this is done in a very experimental way, in so called project-weeks, this is not yet an accepted way of teaching in a school.

A place where TEX could come in handy. Perhaps some teachers could use TEX to produce material for students, but I think this option would be used by a very small minority. No, there is no place were TEX fits into schools. It is too big, too powerful.

Conclusion

TEX is definitely a good tool for typesetting. I don't want to stop anybody from using it. I like to use it myself, in my spare time. But I think that TEX has nothing to do in schools. Let's keep it in the academic and commercial world.

TEX in schools: just say no.

Bibliography

BYTE **7**#8 (August 1982) An all LOGO issue

Harvey, Brian *Computer Science LOGO Style* 3 Volumes. Cambridge, Mass.: MIT Press. (typeset with TEX)

Knuth, Donald E. *The TEXbook*. Reading, Mass.: Addison-Wesley, 1984.

Knuth, Donald E. *TEX: The Program*. Reading, Mass.: Addison-Wesley, 1984

Lovis, D. and Tagg, E.D. *Computers in Education* Proceedings of the IFIP TC 3 Conference ECCE Lausanne 88. Amsterdam 1988: Elsevier Science Publishers

Papert, Seymour *Mindstorms: Children, Computers and Powerful Ideas*, Harvester Press

Appendix

Programming Examples

Compare the following two listings which compute prime numbers. The second one is taken out of *The TEXbook*, the first one was written in LOGO in maybe 5 minutes[1].

The only thing to understand in the LOGO program is the concept of lists: they are like groups in TEX or if you want, like LISP lists, but with brackets instead of parentheses. And FPUT puts a given element into a list at the first position. With this, you should be able to read the program.

```
TO PRIMES :MAX
 PRINT PRIMELIST MAKELIST 2 :MAX
END

TO PRIMELIST :LIST
 IF EMPTYP :LIST [OUTPUT []]
 OUTPUT FPUT FIRST :LIST PRIMELIST REMOVEMULTIPLE (FIRST :LIST) :LIST
END

TO REMOVEMULTIPLE :BASE :LIST
 IF EMPTYP :LIST [OUTPUT []]
 IF (REMAINDER (FIRST :LIST) :BASE) = 0
        [OUTPUT REMOVEMULTIPLE :BASE (BUTFIRST :LIST)]
 OUTPUT FPUT FIRST :LIST REMOVEMULTIPLE :BASE BUTFIRST :LIST
END

TO MAKELIST :START :STOP
 IF :START > :STOP [OUTPUT []]
 OUTPUT FPUT :START MAKELIST (:START + 1) :STOP
END
```

Now, to print the primes up to a given number, you just say PRIMES *number*.
And now, for the same thing in TEX:

```
\newif\ifprime \newif\ifunknown
\newcount\n \newcount\p \newcount\d \newcount\a
\def\primes#1{2,~3% assume that #1 is at least 3
  \n=#1 \advance\n by-2 % n more to go
  \p=5 % odd primes starting with p
  \loop\ifnum\n>0 \printifprime\advance\p by2 \repeat}
\def\printp{, % we will invoke \printp if p is prime
  \ifnum\n=1 and~\fi % this precedes the last value
  \number\p \advance\n by -1 }
\def\printifprime{\testprimality \ifprime\printp\fi}
\def\testprimality{{\d=3 \global\primetrue
  \loop\trialdivision \ifunknown\advance\d by2 \repeat}}
\def\trialdivision{\a=\p \divide\a by\d
  \ifnum\a>\d \unknowntrue\else\unknownfalse\fi
  \multiply\a by\d
  \ifnum\a=\p \global\primefalse\unknownfalse\fi}
```

Called by \primes *number*. Talk about legible programs!

---

[1] Thanks to Erich Neuwirth for the program, he cooked it up in that amount of time.