DOCUMENT RESUME

ED 264 718 FL 015 372

AUTHOR Paillet, Jean-Pierre
TITLE Computers in the Teaching of Linguistics.
PUB DATE 84
NOTE 11p.; In: Carleton Papers in Applied Language Studies, Volume I (FL 015 370).
PUB TYPE Viewpoints (120) -- Reports - Descriptive (141)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Advanced Courses; Classroom Techniques; Computer Assisted Instruction; *Divergent Thinking; Higher Education; Information Processing; Interviews; *Language Processing; *Linguistics; *Programing Languages; Research Methodology
IDENTIFIERS Carleton University ON; *LOGO Programing Language

ABSTRACT

In an interview, a professor of linguistics at Carleton University (Ontario) discusses his use of computers and the programing language LOGO in a fourth-year linguistics course. LOGO was chosen because of its similarity to natural language and its method of structuring data. The first use was in an experimental linguistics seminar, in which the computer was used to expose students to a way of thinking relevant to language and to its use as a tool for conducting linguistics experiments. The computer use was found to be successful in promoting interactive learning and divergent thinking, and in getting across several basic concepts to be used in later courses. Use of computers and of LOGO in education in general is supported because of their ability to promote divergent rather than convergent thinking, and development of a game grammar with even more flexibility than LOGO is seen as a next step. (MSE)

Computers in the Teaching of Linguistics

An Interview with

Jean-Pierre Paillet

Department of Linguistics
Carleton University

Q. Professor Paillett, for the past couple of years now you've been using the computer in fourth year courses in Linguistics at Carleton University. Do you want to talk a little bit about how you were using the computer in these courses?

J-P. P. Yes. I started working with LOGO by myself and becoming acquainted with the basic principles of it, and so on. Over the summer of '82 I decided it was time to do something a little bit more practical. So I arranged to use the Experimental Linguistics seminar to acquaint students with LOGO. The point of the course was not to teach them how to program, however, but rather, first to expose them to a way of thinking which I thought was relevant to thinking about language and second, to expose them to an instrument which could be useful in doing experimental linguistics. The difficulty in talking about this is always the same, mainly that with regards

to language LOGO or any other kind of language is both the instrument for doing experiments and the model of the object. That is, the structure of LOGO in many ways is the same as the structure of natural languages. At the same time one can use LOGO in a very efficient fashion for doing experiments on syntax and semantics. So, in the first term that I used it, the emphasis was on getting students to practice a certain way of thinking, what one might call procedural thinking, as opposed to other kinds of thinking which we don't have names for. This is inspired by Seymour Papert, and the common reference for all that is *Mindstorms* in which he introduces the idea that people develop skills and aptitudes according to what their environment gives them models for. And if their environment doesn't offer many opportunities to become aware of procedural thinking, then you may very well have procedural thinking but you don't have mastery of it because you never have reflexive consciousness of it. And so essentially the idea of LOGO is to provide an environment in which you have to have reflexive consciousness of procedural thinking, in order to practice it in a conscious manner and get mastery of it. And naturally in order to understand how language works one has to have mastery of procedural thinking. So that was the point of the course.

Q. You keep refering to LOGO specifically. How does LOGO differ from other computer languages?

J-P. P. Ah, a good question. LOGO is a cousin of LISP. LISP was the first successful list processing language. Between these and other languages there is a fundamental difference in structure—and naturally in implementation, because the structure determines implementation—and in the relationship of the program-

ming language to the object it works on. In most programming languages, the programs are objects of a totally different nature from what they work on. For instance, if you use FORTRAN or BASIC you have commands that operate on numbers or that operate on strings or other objects. And the commands themselves are not numbers or strings, or whatever; they are a different kind of object. In LISP or LOGO, the representation of the commands is of the same form as the objects they operate on, namely lists. So this introduces a uniformity into the whole way of thinking. One thinks of the data structures, i.e. the objects being operated on, and the control structures, i.e. the programs, in the same way. One can actually have data structures and control structures interacting in homogenous fashion. One of the consequences of that is that the language is not preset. In BASIC or FORTRAN you have a number of basic units which are the available commands and all you can do is string them together in various ways and that's all there is to it. In LISP or LOGO or other list-processing languages, the way you define an operation is by giving a list of sub-operations; each of them is called a procedure. And each sub-operation is itself a list of sub-operations, and so on. The result is the list structure. Now the things you work on also are list structures, so that you think of the two in the same way. Now, I don't know where to go further with that.

Q. Except to ask why that is a particular advantage in terms of what you were working with—natural language. What's the particular advantage?

J-P. P. Ah good. One fundamental property of language is that it has what you might call a layered structure. If you do an immediate constituent analysis,

you recognize at a sort of coarse level of analysis that a sentence is made up of a noun phrase and a verb phrase, or a subject and a verb, or something like that, whatever terms you use. And then when you look at the inner structure of each of these components of immediate constituents, you find that they're organized in the same way, and so on. That is, the organization of any level you choose is of the same nature. Now a list is an ideal representation for immediate constituent structure. And naturally each of the elements of a list itself is a list, and so on, so that there is a direct correspondence between list structure and immediate constituent analysis, for instance. But further than that, when you describe an operation as a list of sub-operations, some of the sub-operations may have exactly the same structure as the whole thing. This is what we call recursivity. Now recursivity is one of the fundamental properties of natural language. From a formal point of view, you can only observe it. From the point of view of asking yourself questions about why language is the way it is, you might wonder, why is language recursive, what function does that serve? Now when you study a list-processing language, you can figure out exactly what function recursion serves in the list-processing language. And then you can report back the results of your inquiry. This sort of thing is right on the surface of LOGO or LISP. It can be implemented in other languages, but it is an artificial thing that you introduce, and therefore you have to think of it. With LISP or LOGO it is something that is presented to you directly and that you can use with it.

Q. To get back to the course, what is your overall assessment at this particular point of the course? What do you think that the students were able to get in

terms of their understanding of language that they would not have been able to get otherwise?

J-P. P. There's nothing that you need the computer to get. So in that sense, the answer to your question is "nothing special". But one of the primary differences is that students had to act, were involved directly, and could gauge the results of their own actions. It is, you might say, an interactive type of learning, instead of an absorptive type of learning. Now this depends very much on the attitude and the initiative of the students. In the first term of last year, fall of '82, I had an extremely active group of students, and they jumped on it and they made it their own. In the fall of this year I haven't had the same kind of success in the sense that the students were not as ready to make the thing their own as they were in the year before. Within the limitation that this brought, I think that I would put across a number of basic concepts, recursion is one, but there are others which are more technical in nature, such as the concept of register, the notion of a unit, and things like that, which I can now refer to in the second term, when I'm doing experimental linguistics, and use directly talking about language using terms which are inherited from computer science. I inform them, say, that linguistics and computer science are the same thing, that the difference is one of method of approach and not one of subject matter. And so I want to import into linguistics all the acquisitions of computer science.

Q. You're obviously very interested in the whole issue of the computer and education. What are your feelings about the way it's being utilized right now as opposed to its potential?

J-P. P. Well, I can't speak specifically, first because I don't

believe I'm very competent and second because it might not be polite; however, on the whole, the computer is being used mainly to take up the tasks that teachers are bored with, and if the teachers are bored with them, imagine the students. So in a way the computer is being used to reinforce the poorest aspects of instruction. The promise and the difficulty with computer education is that the sorts of things the computer could support are very subversive. Further, it is probable that the students who would be faced with those things would be much more competent at them right away than the teacher. So there are social implications and status implications and things like that which are very difficult to deal with.

Q. How about the potential subversiveness of the computer in education?

J-P. P. Well, I would rather refer you to Seymour Papert and the various things he said in a nice and polite, and very suggestive, way. Here is my little contribution. It is the word "education" itself which has been subverted, because most of the things called education are not education, but indoctrination. I understand the etymology of "education" to be 'to draw out'; however, the task given to most teachers is not to draw out, but to pound in! And most education is this practice. So the thing with computer education is that you can use the computer to pound in more stuff, or you can leave it open-ended. I believe the buzz-word is divergent vs. convergent instruction. I've never been able to do anything but divergent learning myself, so I find it very difficult to imagine someone really learning in a convergent fashion. And that's the thing. Even with LOGO, which is only the first bit, the first demonstration of what

computer education could be like—even with LOGO, which is very limited and so on--one cannot grasp the implications within two weeks of introducing it to students. They go all over the place. And this requires a totally different approach and attitude on the part of the teacher, because you can no longer be the one who has the authority, the one who gives the assignments, or anything like that. The teacher can be only a resource person, and, a as resource person, has to be very humble, because within a few days of mastering the rudiments of LOGO, a student can come up with questions that the teacher doesn't have answers to. I think in that sense it's very subversive because it will either be totally suppressed or it will demand of teachers a totally different attitude to their job, and it will accustom the students from the very beginning to expect totally different things from the teacher—not authority, not know-it-all, but openness and ability and willingness to take on something new, and a complete dissolving of subject-matter boundaries. Something that you learn in your mathematics class may become relevant to geography and things like that.

Q. Would you say that there is a major difference between LOGO and other languages in that LOGO encourages this kind of divergence as opposed to other languages?

J-P. P. Yes. Certainly. In BASIC, to mention the reigning model, you have a repertoire of commands which is given to you and that is all you can do. That is exactly the same thing as having those eight subjects—6 periods a day, 8 subjects—and that's the way it is. It's pigeon-holed to begin with. In LOGO, there's no such thing. BASIC defines the type of data that can be operated on before you start. LOGO has only

one type of data—that is lists—and you decide what your lists are going to represent. This is one of the first things my students had to learn last year and I noticed it was a bit baffling to them. I would write on the board something and I would say: "Well, look, this is a tree. Now, what's a tree?" and it took quite a while for it to dawn on them that a tree was a certain kind of list. And now "this is a table," and it took them just as much time to find out that a table was another kind of list and once they got the hang of it after two or three puzzling trials, then they saw everything as lists. The only thing that remained was how best, how most efficiently to represent something by a list and then everything was theirs and they could do what they wanted. I remember a particular case with one of my students ... I had told them "you must come up with your own projects, because I don't want to force you to do things. I want you to know what you want to do." And this student came up and said, "I want to program a Hangman game"; so I said "fine, let's go and do it." It was 2:30 in the afternoon. We finished it past midnight, but we'd gone through all different sorts of projects, all represented as lists, all compatible with each other, and a bunch of procedures connected together in the appropriate way that would transport the information, and the game was actually a Hangman game. It would take a word—you had to put in some vocabulary—and it would give you the usual choice and the usual information and keep track of how you were doing. If you won, it responded to you in the appropriate fashion, challenged you to another game, and then if you lost, it gave you the appropriate scathing remarks. And that took ten hours from scratch, from knowing nothing about how to go

about it ... And LOGO is actually only the beginning. I've been using LOGO because that's what's available, but I see all sorts of "game grammars" as an extension of LOGO. The way LOGO is organized is that at any point it keeps track of how deep it has gotten itself from the starting point. The main fundamental property of LOGO is that it is self-recursive, so that at any particular point, if you give an instruction which names a particular procedure, it stops everything, keeps track of where it was and pretends that the only thing that exists in the world is that particular procedure. When it's finished, it erases all the information that's relevant only to the inside of the procedure, and returns whatever result was required. So that at any given moment, the LOGO system knows only one thing and that is what it is doing now. "Game grammars" work the same way. This is called procedural organization, and anything which can be implemented in an augmented transmission network, or extended finite state automaton, can be implemented in that fashion. This illustrates one important principle in computer science, introduced a few years ago. It's called the principle of information hiding—that is, information should be available only where it is necessary. Otherwise you get things tangled up and mixed up. This is another thing that BASIC doesn't do. Basic has the information all over the place, and it's available everywhere. So that BASIC programs can become extremely tangled. Pascal was the first language introduced which forced information hiding. It forces it at the expense of flexibility. Whereas lists for LOGO make it a natural thing and do not do it at the expense of flexibility. So the implementation of a game grammar will have the same principle at the core, and very often when

I am working with my students, I have flashes of what it would be like if I had a game grammar of a more general type than just LOGO. And that's the direction I want to go in. I could do the same thing in a more flexible fashion, if I had something more general than LOGO, and my thinking on the subject has been the result of the convergence of what I was thinking about grammars of languages and my practice with LOGO.