ISSN 0734-1717

The National LOGO Exchange

Volume 1 Number 4 FORWARD 100! December 1982

Abelson's LOGO Books Are Winners!

If you have been looking for a useful LOGO book to help you get a handle on what LOGO can do for your Apple and you, be sure to get a copy of Harold Abelson's Apple LOGO (for use with the LCSI version) or LOGO for the Apple (for use with the Terrapin and Krell versions).

It is more than just a reference book. While Abelson teaches you something about LOGO, he also shows you what you can do with it.

Assisted in editing by Bruce Roberts and Dan Watt, Abelson has produced a well-organized, clearly written work which explains LOGO's "simple but complex" workings in an easily understandable manner.

The book starts with a look at the basic turtle commands and error messages. Then, it moves on to procedures and modes. Variables, repetition, and recursion are treated next, all with excellent examples to try. His detailed explanation of the recursion process brings light to what many regard as an extremely confusing matter.

Much attention has been given to turtle geometry elsewhere. Abelson chose to devote the majority of his book to non-turtle LOGO work. If you have heard that LOGO is "just turtle graphics," perhaps you should spend some time within these pages.

Numbers, words, and lists are given lots of attention. Example procedures and their development are presented for consideration. Sample quizzes, random sentence generators, and the evolution of a procedure to play and win the game of NIM make objects worthy of study.

The most fascinating part to me is in the advanced use of lists, particularly in the use of procedures as data. The mere fact that LOGO can write and rewrite programs and then run them is almost beyond appreciation. Abelson brings the reader face to face with this boggling concept in such a way that one cannot help but understand.

An easy-to-use glossary and a thorough detailed comparison between the Texas Instruments LOGO and the LCSI or MIT (Terrapin/Krell) Apple version round out the book.

continued on page 8

Brookline Students Hunt LOGO Bugs

by

Jay Sugarman

When I first heard the word LOGO, I thought it must have something to do with the emblem on the Izod clothing line! I couldn't understand how little alligators related to the school curriculum. Maybe if we were located in the Everglades, but certainly not in Brookline, Massachusetts.

Well, much to my delight, I found out that LOGO was the name of a computer language. This was three years ago. What I would like to share with you is how I now see LOGO fitting into the overall curriculum, and how I have come to relate this to my fourth-grade students, as a result of my experience.

To me, the most beneficial and exciting feature about LOGO is its emphasis on the problem solving process. LOGO not only encourages but demands that students become active participants in the learning process. They come to view the computer as a tool, as something they can control and adapt for their own purposes.

By stressing the educational value of the "debugging process," LOGO teachers can help students not only come to appreciate and understand the process of problem solving, but also to develop a comfortable presence around the computer. The students are not intimidated by the computer (as are some adults!), and are able and willing to devote their time and energy to writing and correcting their programs.

HOW TO SOLVE IT

To help my students to conceptualize the problem solving process, I begin the school year by talking with them about the four stages of problem solving (see George Polya, How to Solve It, 1945):

1. Understand the problem.
2. Plan what to do.
3. Carry out your plan.
4. Examine the solution you obtained.

continued on page 2

The National LOGO Exchange, copyright 1982 Posy Publications, a part of The Posy Collection, all rights reserved. Published monthly September through May, $25 per year, $45 for two years, mailed FIRST CLASS from Charlottesville, VA. Add $5 per year for addresses outside the United States. Single copy price - $3. Opinions expressed by the authors are not necessarily those of The National LOGO Exchange. Address all editorial correspondence to the editor at The National LOGO Exchange, Post Office Box 5341, Charlottesville, VA 22905. ISSN 0734-1717

Editor.....Tom Lough

Christmas and ABC's

One of the things I always have wanted to do was to teach my turtle how to draw all the letters of the alphabet on the screen, in any size, at any position, in any orientation.

I had visions of providing children with the means of quickly creating interesting patterns and displays of letters and words. News bulletins and short compostions would appear easily. Suddenly, spelling might become more important. Writing might take on a whole new meaning. Creativity in composition could open into meadows unimagined.

A note in a prominent publication indicated that the memory of the Apple II would not provide room for the LOGO procedures required to draw all 26 letters of the alphabet. Initially discouraged, I decided to try to do it anyway.

The task is now complete! It is done. And what fun it is to create letter and word patterns on the screen! (Incidentally, I have been told that other people have also succeeded in this task.)

Imagine your name spelled upside down or in a wheel or other shape! Think of all the spelling activities you could help your children invent!

Here is our Christmas gift to you. If you would send us a self-addressed stamped envelope, we will send you a listing of the procedures to draw all 26 ABC's. Be sure to specify whether you have the LCSI or the MIT (Terrapin/Krell) version of LOGO.

Mail your request to NLX ABC's, Box 5341, Charlottesville, VA 22905.

Merry Christmas from each of us at the NLX! May 1983 be the "Year of the Turtle" for all of you!

FORWARD 100!! ▷

LOGO Bugs continued

After discussing this general framework, I then point out its relevance in each of the major disciplines. In reading, I talk about decoding the author's writing and analyzing the content. For writing, I focus on the writing process: the continual revising and editing that occurs from one draft to another.

In mathematics, I rely on word problems and real life situations and how to apply arithmetic processes. For LOGO, I refer back to the revising and editing process involved when writing and explain how these features also extend to the writing of computer programs.

LOGO BUG HUNT

Specifically, one of the most successful ways I've related this "debugging" aspect, the problem solving process, is by creating programs with deliberate errors and then having the students try to find and correct them. The children enjoy these activities and eagerly create their own programs with "errors" to see if their peers can find and correct them too.

Another activity I've used to convey the debugging approach is to design a maze on graph paper which can be solved by writing simple LOGO commands such as FORWARD 30, RIGHT 90 and the like. I then provide the students with the maze and a solution procedure which contains incorrect directions. It is up to them to reread the program, find the errors, and then write the correct solution for traveling through the maze. Their solution is easily tested by use of a LOGO overlay (see Hawaiian Students Love LOGO Overlays, NLX, November 1982).

While not a specific activity, I stress to my students the value of orally explaining both to themselves and to a friend what it is they are thinking when they are debugging a program. From my experience, this technique helps them become more involved and reflective participants in the problem solving process.

While I no longer associate alligators with the word LOGO, I'm continually reflecting about how it relates to the overall school curriculum. I am pleased with the progress both my students and I have made during the past three years. We just have to keep working at getting the "bugs" out!

▷

Jay Sugarman, Ed. D. Curriculum and Instruction, is a teacher in the Brookline (Massachusetts) Public School System, and participated in the Brookline LOGO Project.

TIPPS for TEACHERS

by

Steve Tipps

Getting Started

All of the "good" educational ideas in the world are worthless unless they are tested in real situations -- which is to say with real teachers, real children, and real classrooms. LOGO is in the early stages of making the transition from an academic idea to the real world limitations of time, money, and energy.

Getting started with LOGO and some of the many considerations in adoption of LOGO as a central part of a computer education program is the topic of this column. The questions and the tentative answers are drawn from the work that Glen Bull and I have done with fifteen fourth grade teachers in the Albemarle County Virginia Schools.

The answers are only tentative because one of the outcomes of the LOGO instructional development project will be a classroom guide. The teachers are engaged in finding better answers to the questions as the year goes along.

WHY LOGO?

Readers of the NLX probably already have their own answers to this question. LOGOphiles will cite as reasons for LOGO:

1. Having the child in control of the computer.
2. Using the computer as a tool to create interesting results in the form of graphics and language.
3. Exploring mathematical and geometric concepts in an exciting, technological environment.
4. Emphasis on problem solving and critical thinking in a context which leads to good programming principles and habits.

In the first and second cases, LOGO is an end in itself. LOGO is also a means to the ends in the third and fourth.

In the continuing debate regarding "computer literacy," LOGO is unique because it provides actual computer competency. Compare LOGO with approaches which focus upon isolated information about hardware and history of computing.

Such items are of little good or merit if the computer isn't being used to create interesting and useful projects. They are just the kind of nonsense being promoted in the name of computer awareness which LOGO should make passe. When children know how to use the computer, understanding some of the whys and hows, RAMs and ROMs, technology and terminology make more sense. But ... LOGO also demands a different kind of teaching.

HOW MUCH DO TEACHERS HAVE TO KNOW IN ORDER TO TEACH LOGO?

The answer is a good deal more than it takes to plug a child into a computer to drill math facts or answer capitals of states. But the goals are also much different and the professional educator will see the computer as a new tool for achieving the higher aims of education -- critical thinking and creativity -- rather than a new delivery system for the lowest objectives -- rote memorization and recitation.

Most needed for teaching LOGO are teachers who are willing to learn. The simple commands which allow almost immediate access to control over the turtle are somewhat deceptive.

The commands comouflage certain very sophisticated programming concepts -- control, repetition, levels of interest, variability, generality, modularity, extensibility. These concepts are not to be defined but experienced as the teacher becomes a learner again.

The same mistakes and struggles which the teacher goes through will make them more understanding and responsive to the frustrations of learning. Remembering when we didn't know how to read or multiply or remembering how we learned is difficult. Working with a programming language, however, is a constant learning experience which starts with the first command.

After the willingness and will to learn are accepted, several different ways to build skills are possible. One way is to learn with the children by keeping a LOGO manual such as Abelson's at your elbow. (See review elsewhere in this issue.)

Trial and error are certainly within the philosophy of LOGO and many teachers and parents and children are going about learning LOGO just this way. However, some of the long-range benefits of learning LOGO may also be slighted in this approach.

When Dr. Carlos Guttierez, Superintendent of Albemarle County Schools, decided with his staff that LOGO would be the basis of the elementary computer education pro-

continued on page 4

gram, he recognized that he could not simply decree, "LOGO!" and leave it. Staff development was an important part of the plan.

Glen Bull and I agreed to work with the teachers in the project to guide them in understanding the programming concepts embedded in the language and in making instructional decisions about implementation.

We already have had two all-day workshops and seven weekly two-hour meetings with the teachers. In that time, the teachers have developed conceptual and language understanding of LOGO by creating many turtle graphics. They also have learned something about terminology and technology in ways that directly affect their ability to create, save, and print their work.

After about twenty hours of instruction over two months, we felt that they had the necessary skills to begin work with children and to begin creating microworlds for themselves. (Teachers themselves are much more confident about the former than the latter.) The concerns and questions which teachers expressed about implementation and the guidelines proposed may be useful in considering your own plans.

HOW MUCH TIME DOES IT TAKE TO TEACH LOGO?

The most limited resource in a school day is time. If LOGO is considered to be a new subject, then everything else must be squeezed. Instead, teachers have been encouraged to think of ways to squeeze LOGO into the day by not having computer lessons per se.

For example, the ideas of distance and direction are just as appropriate in map study and mathematics. Patterning can be studied in science and language arts. Building the concepts should not be done in isolation from the regular curriculum.

However, three kinds of time need to be planned for: instructional time, independent time, and consulting time.

In instructional time, the teacher would guide the concept or command which is needed by most of the children to get started or progress with LOGO. Instructional time should be very limited. Group times such as those typically found first thing in the morning, before or after lunch, and just prior to departure should be capitalized on rather than a set LOGO lesson time. This time could also be used for sharing of projects.

The focus of the project is independent time in which the children will work on the computer in pairs (or triplets) to develop understanding of skills and invention of projects or applications. The teachers have analyzed the daily schedule so that times when students are doing seatwork, learning centers, or other independent work, they also can be scheduled on the computer.

At first, periods may be rather short (10-15 minutes), but later, 30 minute sessions (some solo) would be needed. Our teachers worked out two day cycles around reading and math groups. 80% utilization of the computer throughout the day was the goal.

Consulting time, or over - the - shoulder time, is the most difficult time to arrange because of the demands placed on the teacher. The luxury of looking over the childrens' shoulders and answering questions and challenging thinking may be the most important teaching time in any subject. Thomas Gordon of TET calls it "optimum time." Some teachers have worked out schedules for consulting time early in the morning. How this will be worked out in the long run is not certain.

HOW SOON DO I INTRODUCE NEW COMMANDS? HOW DO I KNOW IF THE CHILDREN ARE REALLY LEARNING ANYTHING?

Our tentative schedule for the year was to introduce turtle and pen commands (concept of control) and stay at that level for the first month to six weeks. Then teachers would introduce REPEAT (concepts of repetition and indirect execution) as the children need it. Procedures (concepts of delayed execution, programming, naming, and invoking) would follow and depend on needs.

Finally, master procedures (concepts of embedding) and modularity) would be introduced. We had not set this as an absolute limit on fourth graders' ability with LOGO. Instead, it was adopted as a tentative goal which would provide enough practice and experience so that more elaborate problem solving and programming in LOGO would have a firm base.

Fourth graders definitely can type more elaborate recipes into the computer using LOGO than are implied in this sentence. However, our rule is that if children cannot read and describe the actions of their program and debug it when there are problems, they are not conceptually sophisticated enough to understand the recipe.

continued on page 8

MICROWORLDS

by

Glen Bull

Languages for Implementing Solutions and Languages for Finding Solutions

A programmer from another planet might be mildly surprised to find that Earth's best-selling program of all time, Visi-Calc (a so-called electronic spreadsheet), was conceived by a novice. It might be said that this was simply beginner's luck. However, I don't think so.

LANGUAGES FOR IMPLEMENTING SOLUTIONS

The characteristics of a tool are most apt to be shaped by the needs of the individual creating it. A computer scientist needs languages with characteristics which enforce programming rigor. In a professional environment the ultimate goal is error-free code. The administrators of a bank are seldom intrigued by the improbable combination of circumstances that caused an account to be off by exactly four decimal places. Neither is the account holder, if the shift is to the left!

Thinking first through a problem tends to decrease the chances of unforeseen errors. This general method of using computers might loosely be said to be a top-down structured approach to programming. Making things up as you go along is frowned upon in circles in which payment is tendered for the end product.

A language characteristic which flows from the need to preplan a program is the variable declaration. A variable declaration is a list of variables which will be used in a program, and is similar to the list of ingredients at the beginning of a recipe in a cookbook.

The origins of the variable declaration are partly due to technologic considerations, as is the case with aspects of most software. A variable declaration also requires the user to think about the program before writing it. In this programming environment the language is a tool which is used to implement a solution which has been planned before the keyboard is touched.

Many aspects of languages designed for introduction of programming to aspiring computer scientists discourage such a trial-and-error approach. These languages, Pascal for example, are compiled. A compiled language makes it difficult to go through a development cycle rapidly since the editor, compiler, and loader may all be separate programs which result in a time delay when they are accessed.

These characteristics improve the efficiency of the language, and hence are due in part to engineering constraints. They also coincidentally penalize the numerous small changes associated with trial-and-error program development. Even if the technologic efficiency associated with compiled languages were not an issue, these features might still be viewed as positive rather than negative to the extent that they encourage good programming practice.

ELECTRONIC SCRATCH PADS

The electronic worksheet provides a different perspective. Visi-Calc is an electronic spreadsheet. It allows numbers to be entered into rows and columns. For example, a household budget consisting of five categories tabulated by month could be created.

	Jan	Feb	Mar	Apr	AVG
Food	175	225	240	180	205
Clothing	60	20	130	50	65
Rent	250	250	250	250	250
Utilities	90	80	75	63	77
Car	30	23	48	19	30
TOTAL	605	598	743	562	627

After the figures are entered, Visi-Calc allows each column to be totaled. Note that the figure at the end of each row is not a simple summation, however. The total amount spent on shelter for the four months was $1000, but the summary at the end of the row is the average ($250) rather than the total amount spent. In this case, the formula of the total amount divided by the number of months has been used to automatically generate the figures in the right-hand column.

Other ways of looking at the figures could be accomplished automatically. For instance, rent and utilities could be combined into a single category labeled housing.

Housing	340	330	325	313	327

The point to be observed is that experimentation is possible without undue penalty. A built-in inflation factor of 10% for utilities could easily be built into a month-by-month projection for the following year simply by multiplying the entire row by 1.1, for example.

continued on page 6

No repetitive calculations are performed by hand. Rather, a formula describing the calculations desired is written and Visi-Calc performs the operations outlined automatically. This ease increases the probability that many changes will be made.

Of course, some forethought invested in the budget categories chosen will be repaid, but an idea sparked by a preliminary set of results can be incorporated easily into the model. The presence of a resident editor makes alteration of the figures themselves easy as well.

A penalty of facilities which enhance ease of use is that resources such as memory are not used efficiently. Many users of Visi-Calc find that a complete application cannot be fully developed on a microcomputer system with 48 kilobytes of memory. Therefore, a spreadsheet may have to be broken up into several pieces handled separately.

A Pascal program could be written which would accomplish the same task with far less memory, and hence it could in a sense be considered more efficient in the use of computing resources - though not necessarily more efficient in conservation of human resources as time. The next generation of microcomputers now being sold with a quarter to half a megabyte of memory may alleviate some aspects of the efficiency problem.

LANGUAGES FOR FINDING SOLUTIONS

Visi-Calc is the equivalent of electronic scratch paper. It allows the user to doodle and muse over the merits of one approach versus another. The very name "Visi-Calc" is a truncation of "visible calculator." There are environments in which experimentation is appropriate. In these environments the computer is used as a tool for finding the solution, as well as for implementing it.

The non-obvious characteristic which Visi-Calc and LOGO have in common is that they both encourage experimentation without penalizing it. This is the advantage of scratch paper. Stray thoughts are not engraved in brass. Sketches on the back of an envelope can be crumpled and discarded. Papers can be shuffled across the desk to endlessly arrange and rearrange a sequence of ideas.

In contrast, a compiled language more closely resembles a bound laboratory notebook in which ideas are more difficult to change once entered.

The power of an idea similar to Visi-Calc might never have been conceived by a professional programmer simply because its structure encourages practices like trial-and-error experimentation which are anathema in a professional application such as development of a company payroll. Someone who earns a living through program development may also have a greater tolerance for non-visible operations and delayed feedback of results.

The needs of the professional and the casual user are not synonymous. Further, the needs of the casual user may be non-obvious to a computer scientist blinded by different needs and more intensive exposure. Only the obvious commercial value of Visi-Calc and variant electronic spreadsheets clearly demonstrated the need for such a tool.

Adults may also have a need for programming languages which encourage experimentation without penalizing it. Such a language would necessarily be both interactive and modular. (Non-modular interactive languages encourage experimentation but quickly produce incomprehensible programs unless prohibitions alien to the natural habits of the novice programmer are imposed by outright force.)

The need for a tool with these characteristics has not yet been generally acknowledged except in some corners of the artificial intelligence community. The demand for such a tool for casual users has been fairly subdued thus far. The lack of resounding cries across the land notwithstanding, concepts such as Visi-Calc are sometimes obvious only in retrospect.

Currently, no interactive, modular computing language is accessible to first-time users across a broad range of different microcomputers. LOGO has these characteristics, but its full capabilities are available only on a few microcomputers at present (such as the Texas Instruments 99/4 and the Apple II).

The increasing number of individuals who are aware of LOGO tend to think of it as a children's language because of its powerful applications in education. In retrospect it may be obvious that adults also have a need for languages for finding solutions, as well as for languages for implementing solutions. ▷

Glen Bull is a professor at the University of Virginia, and teaches LOGO courses at both the graduate and undergraduate level.

LOGO Quilting Party

Have you ever made a quilt? Neither have I. Not a real one, that is. But I have made a LOGO quilt! You and your children can too, by taking advantage of LOGO's ability to piece things together easily.

Remember the now-classic procedure HOUSE? It pieced together the procedures SQUARE and TRIANGLE. And that is just the way a quilt is made: by piecing things together in a set pattern.

This month's activity should help to give your students a better understanding of how complex computer programs are generated: by piecing together simpler ones.

Let's start with a popular quilt pattern called Dutchman's Puzzle. It is made from triangle-shaped pieces of cloth sewed together. Let's learn how to cut out some of these pieces first.

The listing below is a simple procedure to cut out (draw) a patch of the correct shape, that is of a triangle with two equal sides forming a right angle. We will need to know also that the ratio of the length of the long side (the hypotenuse) to the short sides is 1.414, or more exactly, the square root of 2.

```
TO TRIANGLE :SIZE
     MAKE "LEG :SIZE/1.414
     FORWARD :SIZE
     RIGHT 135
     FORWARD :LEG
     RIGHT 90
     FORWARD :LEG
     RIGHT 135
     END
```

Experiment with this procedure a little. Try several different sizes and turtle headings. Make up some of your own patterns with it, using the REPEAT commands and a turn or two.

A quadrant of a patch of the Dutchman's Puzzle has as its main pattern two of these triangular patches joined together with smaller patches of the background color.

Let us concentrate on the two main triangles. The tip of one is placed at the center of the long side of the other. MOVE is a handy procedure to help with this.

```
TO MOVE :SIZE
     PENUP
     RIGHT 90
     FORWARD :SIZE/2
     LEFT 90
     PENDOWN
     END
```

Now we can "sew" a quadrant of our quilt with the procedure QUAD.

```
TO QUAD :SIZE
     TRIANGLE :SIZE
     MOVE :SIZE
     TRIANGLE :SIZE
     RIGHT 180
     MOVE :SIZE
     RIGHT 180
     END
```

Note that we brought the turtle back to its original position and heading. This is very important to do in most repetitive drawing situations you will encounter.

At this point, LOGO's power to synthesize comes into play. Once your children understand how to assemble procedures in this manner, they will be becoming better problem solvers. After all, just about any problem, regardless of its complexity, can be subdivided into smaller less complex subproblems.

Once the subproblems have been divided down enough to become manageable, they can be dealt with one at a time. These "subsolutions" then can be reassembled together to form the eventual complete solution. Making the quilt is a lot like that.

It is easy now to "sew" the first patch.

```
TO PATCH :SIZE
     REPEAT 4 [QUAD :SIZE RT 90]
     END
```

From here, can you figure out the way to assemble a row of PATCHes together? Right! You need another move of some kind. We have called ours HOP.

```
TO HOP :SIZE
     PENUP
     RIGHT 90
     FORWARD 2 * :SIZE
     LEFT 90
     PENDOWN
     END

TO ROW :SIZE :NUMBER
     REPEAT :NUMBER [ PATCH
:SIZE HOP :SIZE ]
     END
```

And finally, to make the quilt, we need only specify how many rows and figure out how to connect them together. NEXTROW is the procedure we used.

```
TO NEXTROW :SIZE :NUMBER
     PENUP
     LEFT 90
     FORWARD [ 2 * :NUMBER *
:SIZE ]
     LEFT 90
     FORWARD 2*:SIZE
     RIGHT 180
     PENDOWN
     END
```

continued on page 8

Tipps continued

Teachers will be challenged to differentiate children who can type in a recipe for a circle from those who can both formulate and manipulate CIRCLE for their needs. Need and understanding are the determiners of what to do and how fast to move with the children.

In only two weeks with LOGO, several of the teachers have found that the children are more facile with controlling the turtle than we had supposed they would be. Children are keeping logs of their computer time and notebooks of their progress for assessment.

Teachers will have to make a schedule which suits their class progress. Because LOGO is not a lock-step curriculum now (and we hope it remains so as we learn more about using it), much freedom and responsibility resides with the classroom teacher.

CONCLUSIONS

Every child should have the opportunity to work on the computer and progress toward improved ability in turtling. The teacher provides direction, time, and support. As in direction, time, and support. As in all learning, the responsibility for learning belongs to the learner.

One last guideline in a LOGO environment is that working on the computer is not a frill or extra which can be withheld or denied any more than a teacher would withhold spelling books from a child who talked out of turn or ran in the hall.

Responsible purposeful behavior is expected from all children who work with the computer and may include excitement and fun, but not silliness or carelessness.

On a recent visit to a computer awareness class, I found about thirty "extremely active" students in a converted storage room with five computers. The teacher warned me to not put down the computer I brought because she could not guarantee its safety!

I do not believe that this situation is typical, but I do believe that the many ways that computer education can go wrong are frightening. When computers are treated as something outside the traditional goals of education (nurturing competent responsible human beings), we have taken a dangerous step. As we are just getting started with LOGO, let's not lose sight of the big goals. ▷

Steve Tipps is a professor at the University of Virginia, and conducts LOGO workshops for teachers throughout the eastern United States.

Quilt continued

Now to sew the whole thing together!

```
TO QUILT :SIZE :NUMBER :ROWS
   REPEAT :ROWS [ ROW :SIZE
:NUMBER NEXTROW :SIZE :NUMBER ]
   END
```

In the August (LOGO) issue of BYTE magazine (pages 106-110), Harold Abelson outlined a method to establish quiltlike patterns using the power of LOGO's ability to nest procedures. If you have advanced LOGO students, this would be an excellent time to introduce them to this stimulating concept.

This LOGO exploration could also be used in conjunction with a social studies unit on early American history, or in a home economics class as part of a quilt design unit.

For ideas of other beautiful quilt patterns to adapt for your own use, I recommend Maggie Malone's 1001 Patchwork Designs (Sterling Pub.) and The Patchwork Quilt Design & Coloring Book by Judy Larson and Carol Gull (New Century Pub.). I would like to thank Paula Lewis of Charlottesville for her helpful consultation. ▷

Abelson continued

Perhaps it was not anticipated that the book would receive the heavy use it is getting. The spiral binding and cardboard covers should be replaced with more durable material. But, this is more of a testament to the worth of the book rather than to a publisher's oversight.

If you have an Apple computer and a LOGO version, do not fail to obtain a copy of this book! Harold Abelson, Apple LOGO (for the LCSI version) and LOGO for the Apple II (for the Terrapin/Krell version), BYTE McGraw-Hill, 1982. ▷

From the NLX Mailbag

We have built on the CONTROL procedure mentioned in the November NLX. The TI LOGO listing below has a few more things added, and has the advantage of not stopping the program if a different key (other than E,S,D,X) is accidentally pressed.

```
TO GUIDE
   MAKE "KEY READCHAR
   IF :KEY = "F THEN THAW
   IF :KEY = "F THEN FREEZE
   IF :KEY = "O THEN SY 97
   IF  :KEY = "P THEN PRINT XCOR
PRINT YCOR
   IF :KEY = "H THEN HOME
   IF :KEY = "E THEN SH 0
   IF :KEY = "S THEN SH 270
   IF :KEY = "D THEN SH 90
   IF :KEY = "X THEN SH 180
   GUIDE
   END
```

Elaine Blitman
Punahou School, Honolulu HI