LOGO AS A LANGUAGE TO TEACH NON-MAJORS THE ESSENTIALS OF PROGRAMMING

by Gregg Brownell
Department of EDCI
Bowling Green State University
Bowling Green, Ohio 43403

Abstract: Departments of Computer Information Systems and Computer Science frequently offer an introduction to computing course for non-majors. One component of the course is usually an introduction to programming. The language choice for this component is critical if instructional goals related to programming are to be achieved. The course is often taught using either BASIC or Pascal. These languages may in fact be impediments to instructing this population of students in the essentials of programming. Logo, as a dialect of LISP, is a modern, powerful, procedural language which is an attractive alternative for the course. It not only is easy for students to begin using, but also offers capabilities such as list processing, recursion and extensibility which can be used to illustrate modern programming practices.

INTRODUCTION

Many computing departments offer an introduction to computing course for non-majors. Such a course generally consists of an introduction to each of the following: computer terminology, basic computer hardware, computer programming, applications software, computer use in society, future trends, and implications of current and projected computer use (3, 14). The course may serve a range of majors from the humanities to sciences. The need for the course stems from the introduction of the computer to many areas of 20th century life. For these students the computer is a cultural object: a dynamic development in their culture which demands understanding. For many, the computer will eventually be a tool on the job for personal use. Few of these students, however, will ever do any programming beyond their experiences in the introductory course. This paper will focus specifically on the programming component of such a course and the way Logo may help achieve course goals related to programming.

Historically, programming is taught to non-majors for several valid reasons, as listed below.

* Engaging in programming projects aids students in understanding how the computer works. Programming a computer can make abstract concepts about computer hardware more concrete.
* Solving problems on a computer helps students to understand the types of problems to which computers can be applied.
* Through programming, students can be introduced to structured programming as one of the important intellectual concepts of this century.
* By actually creating programs, students can be introduced to a second important concept, top-down design, as it relates to problem solving in general and to computer science in particular.
* Students who may never deal intimately with complex technological projects can gain an appreciation of the resources, abilities and training needed by those who do. Such a realization can aid students in their role as informed citizens in a participatory democracy.

LANGUAGE SELECTION

The programming component of such a course is frequently implemented in BASIC or Pascal. Unfortunately, these languages may not best serve the intended population. Although the first programming experience for non-majors must be of substance, it should also be engaging, inviting and designed to illustrate important concepts about computer programming, problem solving and computer use. The language chosen to accomplish this must have certain characteristics. It must, initially,

be easy to use. With little instruction, students should be able to write interesting, meaningful programs with the language. In addition to ease of use, it must be a powerful, modern, procedural language capable of illustrating the essential of programming a computer using up-to-date techniques.

BASIC, especially in its most readily available implementations, does not possess these characteristics. It is easy to start using but it is not as powerful or modern as other languages. It is not procedural and lacks recursion. It is easy for students to write poorly structured programs in BASIC. Indeed, many college instructors spend a good deal of time helping students unlearn bad habits picked up in a poorly taught high school BASIC course. There are, of curse, implementations of BASIC which address some of these problems, but they are not as widespread as some earlier versions which do not. Also, a good instructor can attempt to teach around such limitations, but usually at a considerable cost of time and effort both for the student and the instructor. This time and effort can best be spent in other ways.

Pascal, on the other hand, is a powerful, procedural language, but is not easy to begin using (12). Pascal is usually the language of choice for introducing majors to programming principles and is an excellent language for such a population. It can, however, be confusing to non-majors who tend to have varying levels of ability. Such students, unlike majors, do not need a subset of Pascal as a prerequisite to more programming courses, since they are quite likely to never program again! What they need is a powerful, friendly language that can serve as an invitation to an understanding of some basic concepts about programming.

LOGO - ANOTHER POSSIBILITY

Logo is a dialect of LISP developed at the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology by Seymour Papert and his associates. It is a modern, powerful programming language. It is perhaps best known for its turtle graphics environment which is often used to introduce young children to the computer. Logo is, however, much more than a language used with young children. In fact, it is used to teach physics and geometry at the college and high school levels (6, 10). It has also been used as the language of choice in introductory computer science courses (5). Indeed, an entire series of texts which use Logo to illustrate computer science concepts has been written by Harvey (7,8,9). Although easy to begin using, it is a modern language that offers, among other things, sophisticated list handling capabilities and recursion. Several of Logo's important characteristics are discussed below.

Students who may never deal intimately with complex technological projects can gain an appreciation of the resources, abilities and training needed by those who do.

Logo is Powerful Yet Easy to Use

Logo offers a low threshold and a high ceiling. This means that Logo is very easy for students to get into, but that the power of Logo allows students (and instructors) to go as far as they like with it. Papert likens this to a native language such as English, which is used by young children, but is also used to produce literary masterpieces (13). Non-majors need such experiences. They need immediate, meaningful successes in programming, but they also need those experiences to be in a language that allows them to grow toward a more complex understanding of programming.

Logo is Procedural and Extendable

Logo procedures may be written and debugged independently. Each procedure may stand alone as a separate program. Procedures can be used as subprocedures within other procedures and within superprocedures (a superprocedure makes use of subprocedures and is the highest level procedure in the program). A student using Logo is guided toward seeing complex problems as a series of smaller modules by the very structure of Logo. By designing superprocedures, creating and debugging subprocedures and then combining the subprocedures to form finished programs, students become acquainted with top-down design. Also, procedures can be given meaningful names and can be used to extend the language beyond the Logo primitives contained in the Logo interpreter. In this way, Logo is extensible. The student extends the language by creating new Logo commands.

Logo Offers Turtle Graphics

Turtle graphics is an environment where students can program an object, the turtle, to move across a screen and draw geometric shapes. For many students, initial programming principles can be made concrete by directing the turtle to create familiar objects and variations of familiar objects. The fact that students can generate geometric figures and even act out, kinesthetically, the turtle's motions, helps to make abstract programming concepts concrete. This, in turn, makes the abstract more understandable. Introducing students to programming in this way can lead to a sound understanding of basic programming principles and can be the groundwork for introducing more advanced ideas.

Logo Offers List Processing

The major way that Logo groups data objects together is through the use of lists which, unlike arrays, are dynamic: may get larger or smaller during program execution. Additionally, lists offer greater flexibility than arrays in handling data. The elements of a list may be any Logo object: a number, a word or another list. Also, lists are a natural way of creating hierarchical data structures. This allows the creation of structures which are comprised of objects which are themselves composed of objects (2). Logo's list handling ability allows students to experiment with text manipulation in interesting ways.

Logo is Recursive

Recursion is an important concept in computer science. Recursion is important because it allows a problem to be restated in terms of itself. This allows compact solutions to complex problems. Logo allows recursion and can be used to illustrate this important concept. For an example of Logo's recursive capabilities as used to solve the classic Tower of Hanoi puzzle, see Harvey (6).

CONCLUSION

There is a real need to introduce non-majors to the essentials of computer programming. This introduction sometimes fails because students are presented with an inappropriate computer language. Such languages can act as impediments to understanding the important aspects of programming and top-down design. To accomplish the necessary goals related to programming when working with non-majors, the instructor needs a relevant, engaging, powerful tool. Logo is such a tool. By providing all the essentials of a modern programming language in a form readily accessible to students, Logo is a viable alternative in such a course. As such it is a good match for students who need to know about programming, but who do not intend to work in the computer field.

REFERENCES/FURTHER READING

1. Abelson, Harold and Andrea diSessa. Turtle Geometry: The Computer as a Medium for Exploring Mathematics. Cambridge, MA: MIT Press, 1981.
2. Abelson, Harold. "A Beginner's Guide to Logo." Byte 7, no. 8 (1982): 88-112.
3. Bowling Green State University 1987-1989 Undergraduate Catalog, Bowling Green, OH, 1987, p. 166.
4. billstein, rich, Shlomo Libeskind and Johnny W. Lott. Logo. Menlo Park, CA: The Benjamin Cummings Publishing Company, 1985.
5. Giangrande, Ernie and Peter Allmaker. "An Introductory Computer Science Course Using Logo: A Case Study." Proceedings of the National Educational Computing conference 1988, annual conference, Dallas, TExas. Eugene, OR: International Council on Computers for Education, pp. 304-307.
6. Harvey, Brian, "Why Logo?" Byte 7, no 8. (1982): 163-193.
7. Harvey, Brian. Computer Science Logo Style Volume 1. Cambridge, MA: MIT Press, 1985.
8. Harvey, Brian. Computer Science Logo Style Volume 2. Cambridge, MA: MIT Press, 1985.
9. Harvey, Brian. Computer Science Logo Style Volume 3. Cambridge, MA: MIT Press, 1985.
10. Lough, Tom. "Logo and Physics." in Cannings, Terence R. and Stephen W. Brown. The Information Age Classroom: Using the Computer as a Tool. Irvine, CA: Franklin, Beedle & Associates, 1986, pp. 200-206.
11. Lukas, George and Joan Lukas. Logo: Principles, Programming, Projects.
12. Masterson, Fred A. "Languages for Students." Byte 9, no. 6 (1984): 233-238.
13. Papert, Seymour. Mindstorms: Children, Computers and Powerful Ideas. New York: Basic Books, 1980.
14. University of New Hampshire Undergraduate Catalog 1986-1987. Durham, New Hampshire, 1986.

AUTHOR'S BIOGRAPHY

Gregg Brownell is an assistant professor of computer education at Bowling Green State University in Bowling Green, Ohio. He has worked as a programmer, taught and designed CIS undergraduate curriculums and worked with teachers on computer education in the schools. He is the author of Computers and Teaching (West Publishing Company, 1987).

Information Systems & Computing
Academic Professionals

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096