The role of computer programming in education

Moderator:
Ken Kahn
President
Animated Programs
49 Fay Avenue, San Carlos
CA 94070 USA
kenkahn@toontalk.com

Summariser:
Harriette L. Spiegel
Florida State University, USA
hls4522@garnet.acns.fsu.edu

Discussion Schedule
Discussion: 6 -13 September 1999
Summing up: 14 -15 September 1999

Pre-discussion paper

The 1960s saw the birth of Logo and Basic, programming languages designed for use by all students. The idea was that computer programming could play an integral part in all education. Researchers began exploring how programming could be integrated into courses on math, physics, biology, and engineering. But it wasn't only hard sciences and engineering: proponents of computer programming wrote about how the study of history, foreign languages, art, music, and much more could be enhanced by integrating computer programming. Some research tried to explore how very young children (5 or 6 years old) could benefit, while other research focussed on college-level math or physics.

Such efforts continue today (e.g. http://www.logofoundation.org) but they peaked in the mid-1980s and have been declining ever since. Recently, however, new programming languages and systems have been developed with many of the same goals as Logo or Basic. These new languages are:

1. ToonTalk (http://www.toontalk.com) - an animated game-like world for constructing, debugging, and running concurrent programs.
3. LEGO Robotics Invention System and RoboLab (http://www.legomindstorms.com) - two very different visual languages for controlling LEGO's motors, lights, sensors and other robotic components.

Questions for discussion include:

1. Was the original vision behind Logo (best articulated by Seymour Papert) flawed? Or was the vision fine but the technology inadequate? Or were the problems more societal and political?
2. Why did interest in programming in school drop in the last 10 to 15 years?
3. All of the new programming systems include many components to aid learning to master the programming language ranging from interactive tutorials to animated demos to quasi-intelligent guide or coach characters. Is this what was lacking before?
4. All of the new languages are visual programming languages where text plays only a minor role. Will this make a significant difference?
5. Some of the new languages (those from LEGO and K'NEX) are specialized for robotics programming. Is this a good thing? Does this limit their scope?
6. Is computer programming just too difficult? For students? For teachers?
Post-discussion summary

The discussion addressed six initial questions:

- Was the original vision behind LOGO (best articulated by Seymour Papert) flawed? Or was the vision fine but the technology inadequate? Or were the problems more societal and political?
- Why did interest in programming in school drop in the last 10 to 15 years?
- All of the new programming systems include many components to aid learning to master the programming language ranging from interactive tutorials to animated demos to quasi-intelligent guide or coach characters. Is this what was lacking before?
- All of the new languages are visual programming languages where text plays only a minor role. Will this make a significant difference?
- Some of the new languages (those from Lego and K NEX) are specialized for robotics programming. Is this a good thing? Does this limit their scope?
- Is computer programming just too difficult? For students? For teachers?

Topics of discussion ranged from the most prevalent, the issue of the value of programming in the curriculum (especially for elementary and secondary education), to specific computer languages and applications such as C++ and Hyperstudio.

Advocates of teaching programming in the schools enumerated characteristics of programming that are valuable to education: Programming encourages:

- procedural thinking
- problem solving skills through trial and error functions
- creativity
- thinking about thinking
- game programming
- natural language programming
- simulations
- math and science skills
- basic knowledge about computers
- analysis and exploration of science
- logical thinking ability of students

Several discussants commented that programming should be a core course in the curriculum, like mathematics and even writing. Some claimed that students must learn understanding and manipulation of information before beginning to program or author on computers. In programming, it is important to provide models for the students to emulate. Programming has elements of good design that make it a great learning experience: the iterative process towards a solution, the ultimate goal that has many paths and choices on the way, and the mistakes that provide feedback and experience. Several discussants provided plans for integrating programming into the curriculum, including graphic arts and other required subjects. But important obstacles are time to implement programming, and providing student access to computers.

Some comments recommended that programming not be taught to everyone, but only to students who were pursuing programming as a career. One response to this attitude strongly described what a community of practice should be: a combination of veterans and novices, blending school and the real world. Concerns that such programming languages as C++ were too powerful for socially-responsible use by younger students drew responses of wry reminders that children who are interested in learning about programming are going to do so without waiting to be taught, regardless of whether or not values have been developed.

The point-and-click paradox has led to the user’s demand for more features on computers. As computers became easier to use (rather than requiring the writing of programs), the user’s imaginations demanded more and more from developers. Therefore, one solution is to make programs programmable, allowing the user to customize the features of the program.

Those discussants who were less supportive of teaching programming in the schools also described their reservations:

- The spirit of exploration that is inherent in programming is not effective for all students.
Teachers have been inadequately trained to teach programming.

Because of the development of computer software, there is no longer a need to know how to write programming in order to use the computer.

More interesting activities that are directly relevant to curriculum are possible with modern computers.

With the development of tools such as Excel and Stella that enable model-making and dynamic modeling, programming must add something extra of value.

The self-learnability of the new languages (that are designed to work without teachers) may actually be an obstacle to adoption of the new languages.

The new languages are providing a missing ingredient in the schools, visual-based education--something that pure programming did not have.

The discussion regarding training teachers to teach programming provided several useful ideas:

- Teachers should be trained to guide discovery, systematically, among students.
- Teaching of programming should add fun to learning.
- Teaching of programming should encourage interactive and talkative learning.
- Although the syntax of LOGO (a typical textual programming language) is considered more complex for students up to about the sixth grade, programming should be taught to older students to encourage general skills of problem solving, learning and thinking. Opinions were split as to whether programming languages with a visual syntax will simplify things, especially for younger students.
- The potential of programming to affect metacognitive and self-regulation skill development should be explored by researchers (through the exploration/discovery/learning aspect of programming).
- Teachers should ‘teach,’ not just ‘tell.’
- Papert (http://connectedfamily.com) provided important additions to the 3 R’s (reading, ‘riting, ‘rithmetic): the three X’s of eXploring, eXpressing and eXchanging.

Interesting analogies were made. The development of computer programming was compared to the development of auto mechanics (the need for the user to know how to program or fix the auto is no longer there, leading to the development of specialty careers, rather than the demand for general knowledge). The computer was compared to the pencil (the use to which either is put is important, but few need to know how to make either). However, unlike that of an automobile, the potential of a computer is limited only by human intelligence and imagination. And if the computer is as ubiquitous as the pencil, then its programming is as widely applicable to many human endeavors, again limited only by human imagination.

Several comments were made regarding newer programming projects such as Microworlds, and new languages from LEGO and K’hex. The specialized use of programming, such as in robotic or speech simulations, was considered useful in reaching certain students, but it was emphasized that programming will not solve all problems. Many pointed out that while programming does not appeal to all students, it can have very broad appeal because programs can be constructed in so many areas of interest.

Hyperstudio was an often-addressed topic. The consensus was that it was not true programming, but provided motivation to students. If a teacher provides structure, examples, and guidance, Hyperstudio is a tool to teach organization of content, and use of graphics and hypertext tools. Furthermore, Hyperstudio provides the student, like the LOGO programming language, ownership of the learning environment. (There was also discussion of the availability of LOGO as a programming language from within Hyperstudio, although the feature was designed poorly and is rarely used.)

A distinction between programming and media creation was discussed. Rather than looking at a picture produced with a current paint program as just a picture (or multimedia as simple entertainment), media creation on the computer requires the user to talk about representations, to search and organize options for replication and editing. These characteristics point to the need to teach programming.

Game programming was a favorite topic, described as crossing traditional curriculum lines, encouraging learning. Rather than forbid games (as a waste of time) on the computer, the teacher should insist that the students create their own games, and ensure that the necessary books, examples, tutorials and tools are provided.

Computer projects that are as valuable as programming in their potential for intellectual development were the creation of comic books, the creation of web pages and presentations with paint and animation.
Interested readers of this summary are encouraged to refer to the pertinent discussion or summary for enlightening references, including websites, books, articles and interviews, and authors' names.