Fourth Grade Computer Science

Daniel Frost
Donald Bren School of Information and Computer Sciences
University of California, Irvine
Irvine, CA 92697-3425, USA
frost@uci.edu

ABSTRACT

We describe a module, or sequence of lessons, that has been successfully used to teach basic elements of computer science to fourth grade students. The module was designed to reflect a firm grounding in computer science, to be age-appropriate, to be easily installed in schools, and to support a range of teachers. Over 300 students in grades three through six have taken this module or a related module. The programming language used is a modern variant of Logo called VVLogo, which students access through a Java applet on a web page.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: *computer science education, curriculum.*

General Terms

Design, Languages.

Keywords

K-6 Curriculum, Fourth grade computer science, Logo, VVLogo.

1. INTRODUCTION

Why isn't computer science taught in fourth grade? Science is part of most fourth grade curricula, taught either by the regular classroom teacher or by a visiting science specialist. In California, for example, the fourth grade content standards include topics from Physical Sciences (circuits, magnets), Life Sciences (the food chain, ecosystems), and Earth Sciences (rocks and minerals, erosion) [2]. Why not computer science?

The question has many answers. ACM's *A Model Curriculum for K-12 Computer Science* [9] identifies insufficient teacher preparation, a lack of state-level content standards, and the need for the development of curriculum materials. Another important factor is the common perception that computer science is too hard or too advanced for younger students. When computer science is viewed as akin to advanced math or electrical engineering, this reservation makes sense. However, in this paper we argue that basic concepts of computer science are appropriate for K-6 students in general, and fourth graders in specific, just as are the foundations of other sciences. In fact, the fascination today's children have with computers can make them more receptive to computer science than to, say, geology.

This paper describes a computer science module, lasting 45 minutes a week for about 10 weeks, which has been taught over the last four years at a public elementary school. ("Module" means a sequence of lessons on a specific topic.) More than 300 students, mostly in fourth grade but also in the third, fifth, and sixth grades, have taken different versions of module. The module teaches computer programming (using variants of Logo and Basic) and computer science topics such as abstraction, ethics, computer graphics, hardware, and history. We specifically describe the module as taught to two fourth grade classes in Spring, 2006.

2. DESIGN PRINCIPLES

In designing and delivering the module, the overriding goal was to make every student's experience a positive, educational, and memorable encounter with computer science. The ACM Model Curriculum [9] provides an extensive list of skills and outcomes related to computer science and appropriate for K-8 students. To complement and instantiate the ACM Model Curriculum, the module was designed with four criteria in mind: firm grounding in computer science; age-appropriate materials, concepts, and delivery; use of accessible hardware and software technology; and support for teachers delivering the module.

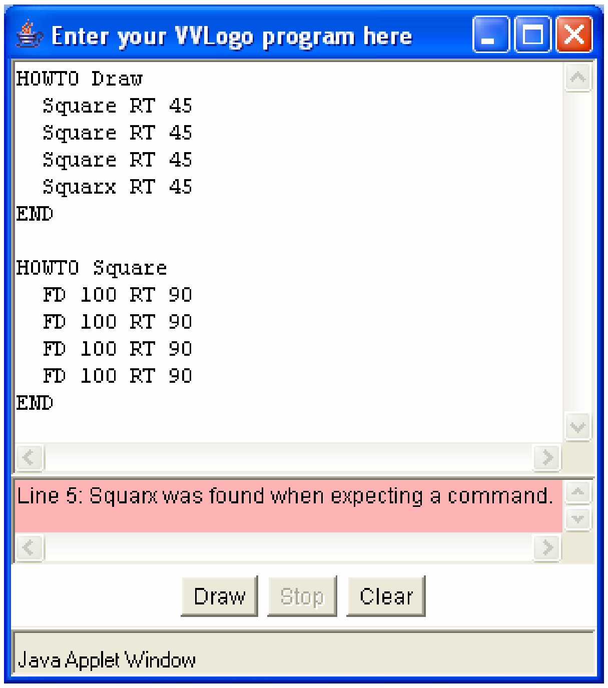
Firm grounding in computer science. Computer science is a broad field which includes the study of computer programming, computer hardware, data structures and algorithms, software infrastructure such as operating systems, networks, and compilers, the design and engineering of software applications, human-computer interaction, and artificial intelligence. In any course only a subset of topics can be covered. For the fourth grade module, we wanted to emphasize the visible, hands-on aspects of computer science, as well as those that relate directly to students' everyday interactions with computers. We therefore designed the module to cover the first elements of computer programming, program design, hardware, computer graphics, and ethics.

It's important to note that the module is not an introduction to computer literacy, and would not be appropriate for a class unless the students had already learned to use the keyboard and had experience with a variety of software applications. (More advanced literacy issues, such as copy and paste, did often come up in the labs and were discussed one-on-one.)

Age-appropriate. Many core concepts of computer science are interesting to, and usable by, fourth graders. Some examples:

- Computers follow programs that are written by people.
- Computer programs are written in languages with syntax and key words.
- Computers have no understanding beyond what is explicitly coded into a computer program.
- For a problem to be solved by a computer, it has to be stated in a completely defined way.
- For information to be used by the computer, it has to be represented as digital data.
- Writing a computer program involves selecting or creating algorithms and data structures.
- Algorithms implemented as computer programs use control structures, such as FALL-THROUGH, IF, REPEAT, and PERFORM-SUBROUTINE.
- A computer programmer has to be able to step through a program, following each instruction, just as the computer does.
- Programs often don't work correctly the first time; much revision and testing is usually needed.
- Different parts of the computer have different functions, dealing with input, output, storage, communication, and computation.
- Computer displays (including printed output) consist of pixels, each of which has a color.
- Communicating via a computer (e.g. with e-mail) means that many copies of a message may exist.

In addition to issues of content, the module emphasizes creativity, experimentation, and understanding in age-appropriate ways. In the computer lab, the students were given guidance and specific tasks, but also allowed to explore and share with each other interesting discoveries. Fourth grade students are learning to generalize concrete experiences with abstract concepts, and to divide large tasks into sub-tasks which are easier to accomplish, both activities were emphasized in the module.



```
HOWTO Draw
  Square RT 45
  Square RT 45
  Square RT 45
  Squarx RT 45
END

HOWTO Square
  FD 100 RT 90
  FD 100 RT 90
  FD 100 RT 90
  FD 100 RT 90
END
```

```
Line 5: Squarx was found when expecting a command.
```

Figure 1: The VVLogo programming window, with a program and an error message.

Use of accessible hardware and software. The good news is that many American primary and middle schools have an ample supply of PCs connected to the Internet. However, these PCs often have a limited set of software. Because of district policies, fear of viruses, insufficient time, and the frequent release of new versions, it may not be easy for a teacher to install additional software. We decided to deliver a programming environment through a Java applet available on a public web site (www.csed.org). As long as the school's PCs have Java-enabled Web browsers, accessing the programming environment is a matter of navigating to the web page. An added advantage is that the language and environment are available to children at home, and to teachers and students at any school.

Teacher-friendly. The module can be taught by classroom teachers, science specialists, parent volunteers (perhaps following the Junior Achievement model [5]), or by faculty and students from a university. The curriculum is flexible and well-supported: all curriculum materials, programming language documentation, handouts, and teacher guides are freely available on the web.

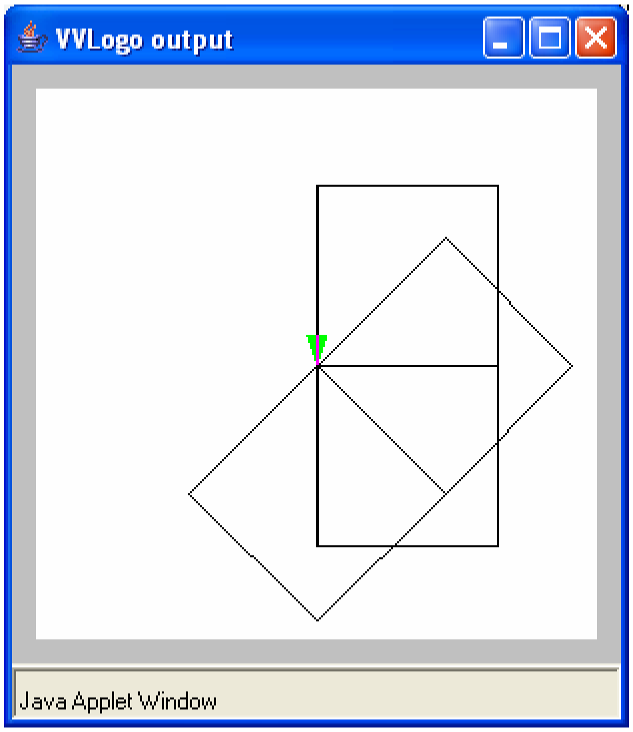


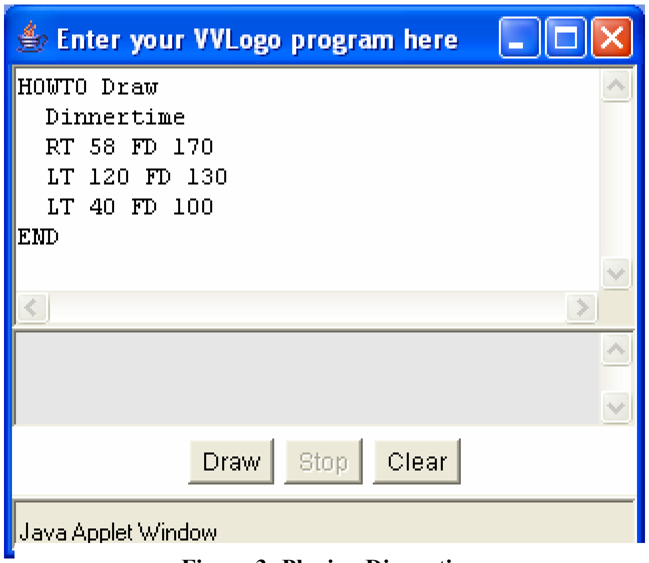
Figure 2: Output from the corrected Figure 1 program.

3. LANGUAGE AND ENVIRONMENT

We developed VVLogo, a version of Logo [8], and VVBasic, a version of BASIC [3]. An important property of a programming language for children is that it allows short—even fewer than ten keystrokes—interesting programs to be written. We focus here on VVLogo, the language taught in the Spring, 2006, module. As in traditional Logo, VVLogo has a turtle, drawn as a green triangle, that can be given the commands Forward (abbreviated FD), Back (BK), Left (LT), and Right (RT), each followed by a number indicating length or degrees. When the turtle moves, its "pen" draws a line. The language is not case sensitive, and uses no punctuation except for parentheses and white space. Classic Logo defines a subroutine between To and End; in VVLogo the clearer (we thought) HowTo is used. A subroutine is invoked by using its name, followed by any parameters. See Figure 1, which produces (after fixing the error introduced on line 5) the output shown in Figure 2. A correct VVLogo program has a main routine named Draw, which is invoked when the student clicks on the button labeled Draw.

After PenUp (PU) the turtle moves without drawing; PenDown (PD) restores the pen. The commands PenColor (PC), TurtleColor (TC), and BGColor (BC) change the pen, turtle, and background colors. Colors are defined using five “primary” colors, Red (R), Green (G), Blue (B), White (W), and Black (K); primaries can be combined with + (addition). For instance, pink is Red + White, and light gray is Black + White + White (or K+W+W). The only control structure we introduced in fourth grade was Repeat, which is followed by a count and then the actions to be repeated in parentheses. For example, to draw a triangle: Repeat 3 (FD 100 RT 120). The Slow command changes the speed at which the turtle moves, turns, and draws; it is very useful for understanding and debugging programs.

The VVLogo environment is accessed on a Web page where a Java applet pops up two windows. The programming window, titled “Enter your VVLogo program here,” has three parts, an editing area on top, an error message area in the middle, and a button area. Pressing the Draw button starts the program; the Stop button (enabled only when a program is running) can cancel a long running program; and the Clear button clears the programming area. Unlike many Logos, there is no command line. The second window, titled "VVLogo output," displays the turtle and the lines it has drawn.



```
HOWTO Draw
  Dinnertime
  RT 58 FD 170
  LT 120 FD 130
  LT 40 FD 100
END
```

Figure 3: Playing Dinnertime.

4. CURRICULUM

The module has evolved and improved each time it has been taught. We describe here the version taught to two fourth grade classes in April, May, and June, 2006. Because of interruptions for statewide testing and field trips, this module was only eight weeks long. Each weekly meeting consisted of about 20 minutes of classroom discussion (with no computers), followed by about 25 minutes in a computer lab. In the lab each student had his or her own computer, and followed, at least to some extent, a one-page set of instructions handed out each week. The module was taught by the author, a parent volunteer, and an undergraduate from the author’s university; in lab three teachers for thirty students is satisfactory.

Week 1. In the classroom we introduced ourselves and administered a pre-test (discussed in section 5). It is almost impossible to explain what programming is, so we talked a little bit about the turtle, gave the students instructions on how to bring up the web page (they each had an account already and knew how to log on), and sent them to the computer lab. By typing in and running some simple VVLogo programs, they became familiar with the editor, the error message area, the turtle window, and the turtle. About a third of the students started experimenting with different parameters and hypothesized commands (if Forward and Back, why not Sideways?). Two or three students in each class thought of a shape or a picture (such as a happy face) and tried to make the turtle draw it.

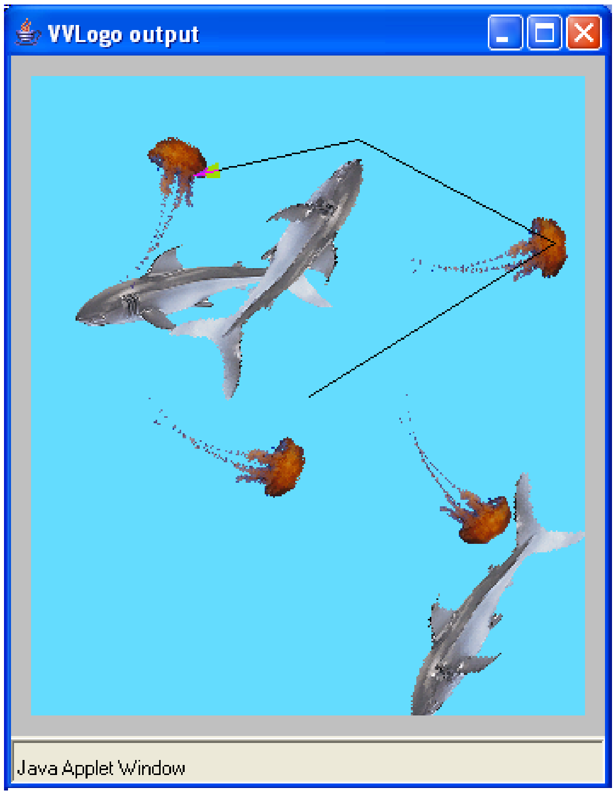


Figure 4: Two jellyfish down, two to go.

Week 2. At the start of each meeting, we always reviewed all previous material, as interactively as possible. Students eagerly volunteered to be the turtle and act out FD, RT, and other commands. We tried as a class to make a student turtle walk in a triangle—this was pretty tricky; it wasn't clear to all that LT and RT are from the turtle's point of view. Also, the students were just learning about angles and degrees in math. In lab, students entered programs using PenUp and PenDown. They also experimented with different sizes of the output window, and observed what happens when the turtle moves too far in one direction (it wraps around).

Week 3. To start, the undergraduate assistant answered questions about life as a college student. (The charm of the undergraduate assistant and his or her rapport with the students is always a factor in the module's success.) The undergraduate assistant invited up a student to be the turtle, and the class tried to guide her back to her seat. She eventually got there. We introduced several topics that we returned to again in following weeks. We talked about the special role of the HowTo Draw, and that new "HowTo"s can be defined and used. In every subsequent lesson we talked about how and when to define HowTos; fourth graders seem to be just on the cusp of being able to understand and use this kind of abstraction. Another topic was error messages; many students had noticed them, but most hadn't understood the connection between the messages and what they had coded. Many students had heard from their friends in higher grades that the colors could be changed, so we introduced PenColor and the five VVLogo primary colors, perhaps a bit earlier than would be pedagogically ideal. In lab, students found that with a HowTo Square they could create interesting patterns. A group of boys was most interested in seeing if they could turn the window black by making the turtle pass over every pixel.

Week 4. We discussed e-mail, with which many students were familiar. After talking about how long an e-mail lasts and how many copies of it are made as it is transmitted, we turned the discussion to safety and ethics. We suggested two rules for students to follow: Never write anything in an e-mail you don't want the entire world to read, and check with your parents before you forward an e-mail. Back to VVLogo, we asked, can we teach the turtle how to (HowTo) draw big letters? The class collaborated on routines for L and T, which they continued working on in lab. The class was told that primary colors can be combined with "+". This was easy for students to grasp, but most were dumbfounded that red plus green equals yellow. In lab they saw it was true, and noticed that the monitor's red and green are actually yellowish versions of those hues.

Week 5. The focus this week was on angles. In the classroom we gave each student a sheet of paper with a large circle on it, and had the students draw in several angles. 45 degrees and 90 degrees were pretty solid. Other angles proved tricky, but the regular teachers were happy because angles were just coming up in the math curriculum. We had a discussion with the class about program readability and good style habits such as not putting too much on one line. In lab students played a built-in VVLogo game called Dinnertime, in which they have to guide the turtle to the tasty jellyfish, while avoiding the turtle-eating sharks (see Figures 3 and 4). Playing this game reinforced students' abilities to estimate angles and distances.

Week 6. In class we examined and discussed the parts of a physical computer. We brought in examples of internal parts such as memory, circuit boards, and disk drives. The categories of input devices, output devices, volatile and static memory, the central processing unit, and the notion of price and speed trade-offs were mostly new to the students and easily grasped. In lab, the students honed their color matching skills by playing a VVLogo game called Secret Agent. The goal is to change the pen color as the turtle traverses four blocks of random colors, so that the pen color matches the blocks and the turtle's movements can go undetected.

Week 7. If a HowTo is defined with the name Square, and then elsewhere you try to use the routine but call it Skware, will the program work? Does the turtle know that Square and Skware sound the same? What if the HowTo is named Square but it actually draws a triangle? The students were evenly divided on these questions, and the ensuing discussion clarified to many students the limits of the computer's intelligence. We also introduced the Repeat command, and in lab the students had a chance to rewrite some previous programs using Repeat, which many found quite interesting.

Week 8. We kept the final class period short, just administering the post test and saying goodbye. In lab, students wrote HowTos named Squiggle containing almost random commands, and then used Repeat to perform multiple Squiggles. For example:

```
HowTo Draw
     Repeat 50 (Squiggle)
End
HowTo Squiggle
     FD 87 RT 113
     BK 23 LT 21
End
```

Interestingly, the results are always attractive no matter how Squiggle is precisely coded; when the module is longer we discuss this phenomenon.

5. EVALUATION

We evaluated whether the module met our goals of being a positive, educational, and memorable encounter with computer science. Our primary tools in the evaluation were pre- and post-tests we administered. In the pre-test we found that every student had access to a computer at home. We asked, "Have you ever sent an e-mail?" Out of 58 students, 34 answered yes (four volunteered that they had only sent one) and 24 answered no. We asked, "What are three words you would use to describe computers?" The top responses are shown in Table 1.

Table 1: Words used to describe computers in pre-test

Words listed by students	**Frequency**
fun	30
smart	15
cool	14
helpful	7
useful	7
interesting	7
other positive words, such as magnificent, exciting, great	26
other negative words, such as hard, boring, virus	11
other neutral words, such as mouse, electricity, research	43

With 106 positive words, 43 neutral, and 11 negative, clearly the fourth graders entered the module with mostly positive attitudes about computers. Another pre-test question was, "Suppose you are on a beach. You are making patterns in the sand by dragging a long stick as you walk. First you walk forward 6 feet. Then you turn left 90 degrees. Then you walk forward 2 feet. Then you walk backwards 4 feet. What shape have you drawn in the sand? What letter of the alphabet does it look like?" 10 students correctly identified the shape as "T"; 9 did not answer the question, and 39 students gave wrong answers, most frequently "J".

In the post-test we again asked students to write three words describing computers. The results were very similar to the pre-test, overwhelmingly positive, with four "boring" or "annoying" responses, and a few responses reflecting the module, such as "input and output." We also gave the students another "drawing in the sand" question, a slightly harder one with the correct answer being "H". 34 students answered correctly, 2 students did not answer, and 22 answered incorrectly, usually "+". The post-test results indicate that about half the students acquired an improved ability to work step by step through detailed instructions.

The pre- and post-tests support our observations that the module was positive and educational for the students. To evaluate the goal of memorable, we have only anecdotal evidence. Fifth graders who took the fourth grade module a year earlier generally have required only one meeting to recall what they had learned. Also, from server logs we can see that the web page continues to be accessed for many months after the last module was taught.

6. RELATED WORK

Logo has long been a popular language for teaching programming to children. It's popularity reached a peak in the 1980's, and diminished afterwards with the wide availability of excellent software for drawing and making music. [4] and [7] are fine collections of articles on Logo philosophy and practice. A list of commercial and public domain implementations of Logo is maintained by the Logo Foundation [6].

An interesting collection of activities which illustrate a wide variety of computer science topics and which are designed for five to twelve year olds is provided by [1]. These activities do not require a computer, which underlines the often elusive observation that computer science is not about computers. The approach presented in this paper is based on the contrasting, but not contradictory, assumptions that computers are accessible and fun to use.

7. CONCLUSIONS

Today, students learn about physics in fourth grade by building a simple compass, learning how to use it, and observing how a nearby magnet affects it. Fourth graders learn beginning earth sciences by recognizing different types of rocks and minerals. In the near future, perhaps all fourth grade students will learn about computer science by writing simple, graphical programs, by learning to recognize the parts of a computer, and by learning how to use computers and software safely and ethically.

Two factors impeding K-6 computer science education are the lack of curriculum materials and the mistaken—we believe—notion that the nature of computer science makes it too hard for younger students. By adopting pedagogical goals and styles used successfully to teach other sciences to K-6 students, computer science can be more widely taught in primary schools. Anyone interested in exploring VVLogo further is encouraged to go to the www.csed.org web page and click on VVLogo.

8. REFERENCES

[1] Bell, T., Witten, I, and Fellows, M. *Computer Science Unplugged*. 2002. Available at http://www.unplugged.canterbury.ac.nz/

[2] California Dept. of Education. *Grade Four Science Content Standards*. URL http://www.cde.ca.gov/be/st/ss/scgrade4.asp, accessed Sept. 6, 2006.

[3] Dartmouth College Computation Center. *Basic*. 1964. Available at http://www.bitsavers.org/pdf/dartmouth/BASIC_Oct64.pdf.

[4] Harper, D. *Logo: Theory and Practice.* Brooks/Cole, Pacific Grove, CA. 1989.

[5] Junior Achievement. *Get Involved Volunteers.* URL http://www.ja.org/involved/involved_vol.shtml, accessed Sept. 6, 2006.

[6] Logo Foundation. *Logo Products.* http://el.media.mit.edu/Logo-foundation/products/software.html, accessed Nov. 15, 2006.

[7] Maddux, D., and LaMont Johnson, D. *Logo: a retrospective.* Haworth Press, New York, 1997.

[8] Papert, S. *Mindstorms: children, computers, and powerful ideas,* second ed. Basic Books, New York, 1993.

[9] Tucker, A., Deek, F., Jones, J., McCowan, D., Stephenson, C., and Verno, A. *A Model Curriculum for K-12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum Committee.* The Association for Computing Machinery, New York, 2003.