Divide and Conquer:
Teaching Programming from a Visual Perspective

Paul Palazolo[1] and Anna Phillips-Lambert[2]

Abstract

Although most undergraduate engineering students are familiar with a variety of different computer applications, many students encounter difficulties with introductory programming. A review of available literature suggests that programming concepts are more easily understood when visually-based explicit examples are introduced as examples of programming language, and this research tests that conclusion.

A sophomore-level civil engineering class at the University of Memphis combines traditional programming concepts with a series of non-traditional pedagogical approaches with the goal of making programming more palatable for the students.

In an effort to make the introduction of programming constructs such as Do-While, If-Then-Else-EndIf, and other syntax independent concepts less abstract to the students, the Legos Mindstorms system was used as an introductory programming system. The Legos Mindstorms system is a robotics system based on the Legos building concept along with a simple icon based programming interface that allows the development of autonomous robotic vehicles and systems. The Legos system allowed for immediate feedback for programs through the performance of the robots. Logical development of the programs is emphasized over syntax through the simplified syntax used in the icon-based programming language. Problems presented reward good logical development through better performance of the robots. The students then move into a Visual Basic programming environment.

While there is no longer a direct physical response to programs, Visual Basic provides for immediate visual feedback to the programs. Students are encouraged to make the transition from the Mindstorms block program structure to a flowchart type logical structure before beginning to develop their code. Nuances of Visual Basic syntax are presented and mapped to the Mindstorms experience.

Both qualitative and quantitative assessment instruments were used throughout the course to provide students' feedback about the experiences and to measure proficiency in programming. This was compared to previous classes taught with the more traditional methods. Results indicate that the previous reluctance with which students approached the programming class has been reduced significantly. Results also indicate that there is a deeper commitment to utilize programming to expand the computer functionality within their professional work. Competency in programming has also increased with the separation of logical development from syntax.

Introduction

The programming class is the third class in a sequence of four classes that make up the Civil Engineering Foundation Sequence. The Foundation Sequence was originally developed to integrate problem solving and communication skills into an existing traditional introductory sequence of surveying, computer usage, and programming. The first two courses in the sequence are design-oriented with student team projects emphasizing problem solving using the tools of civil engineering. The first two courses have been successful

[1] Department of Civil Engineering, The Herff College of Engineering, The University of Memphis
[2] Department of Civil Engineering, The Herff College of Engineering, The University of Memphis

in introducing new students to the civil engineering profession and in significantly enhancing their communication and group work skills based on the students responses and responses from faculty in upper-division courses.

The third course in the sequence, the programming course, had the least amount of modification from the traditional coursework. Typically, it had been presented as a programming course with emphasis on the utilization of programs to solve tedious numerical problems. With the increasing power of spreadsheets and specific problem solving packages such as MathCad, and with the feedback from practicing engineers, the utilization of code developed by the students appeared to have less and less utility for their professional careers. At the same time, the presentation of the course was such that the emphasis on problem solving and communication was not carried over into this course.

The course was originally taught as a FORTRAN programming course. In an attempt to make the course more student friendly, four years ago the course was shifted to Visual Basic. One semester, the course was linked to previous coursework through the use of Visual Basic for Applications in an attempt to show students the extensibility of Excel. Each of these approaches relied on teaching students both the logical development of programs and the particular syntax of the language chosen.

Students and members of the faculty were continually questioning the relevancy of the course. The students were not taking the programming skills that they had developed into the upper division and when asked to develop a computer program, they were often unable to perform the tasks assigned. Most often, the instructors modified their computational assignments so that they could be completed using a spreadsheet or some other computational tool.

From this background, we began to review the materials and methods used in this course. Our main focus was on the integration of this course into the Foundation sequence with the course having the same problem solving and communication components as the first two courses in the sequence. During summer programs working with middle-school children, we had noticed that they were able to understand logical programming concepts when presented with a very simple visual method for developing programs utilized by the Legos Mindstorms system.

In addition to the visual development environment, we had utilized problems with the middle-schoolers that required the selection of a solution from a number of alternative solutions through testing and evaluation. Each solution had a physical component apart from the programming component and emphasized the connection between program and action. Based on these two properties of the Legos Mindstorms system, the visual development of the program logic and the physical problem-solving component, the decision was made to utilize the Mindstorms system in the programming class.

Pedagogical Background

Our primary pedagogical approach in design and implementation of the programming course is a combination of constructivist-based psychology and student-based self-regulated learning. The constructivist approach is a theory of knowledge acquisition that originates from the work of Piaget and Vygotsky, two of the most prominent theorists in educational/developmental psychology. Constructivist-based educational practices encourage students to be active learners and to seek solutions for themselves as opposed to traditional teaching methods where students “learn” by imitating the teacher or by demonstrating minimal competency levels through testing.

An interesting parallel between the psychology of learning and learning to program is found in Vygotsky's primary contribution to constructivism. According to Vygotsky's work, constructivism is based on the theory that thought development is determined by language, yet the language itself is irrelevant as long as the learners share the same constructs. According to this theory, language/speech, which later becomes internalized thought, involves a process of collaboration in the learning community. Consequently, the emphasis in learning in constructivist-based learning environments is targeted towards teaching students to learn content through direct experience with materials and tools where the students themselves construct

knowledge based on collaborative learning strategies and subsequent applications. By using the Legos as materials and tools to solidify programming constructs, and by continually adapting feedback from experimental results and integrating comments from fellow learners, our students created their own collaborative learning environments.

Implementation

The utilization of the Mindstorms system would be approached from two directions. The first would be to develop a series of problems with open-ended solutions that could be used to continue the illustration of the engineering problem solving process that had been begun during the first two courses in the sequence. This would also allow a continued emphasis on communication. The second would be to focus on the logical development of solutions without the requirement for a parallel development of a sophisticated syntax.

The Mindstorms system may be programmed using an icon based visual interface. The command structure is such that there are no keywords or syntactically challenging structure necessary to develop a working program. Programs are developed by dragging program blocks onto a workspace and then connecting these blocks together to form the program structure. Logical control structures and repeat loops are self contained with both the branching elements of the logical elements and the terminals of the repeating structures an integral part of the blocks as they are brought onto the workspace. An example of both of these structures is shown in Figure 1.

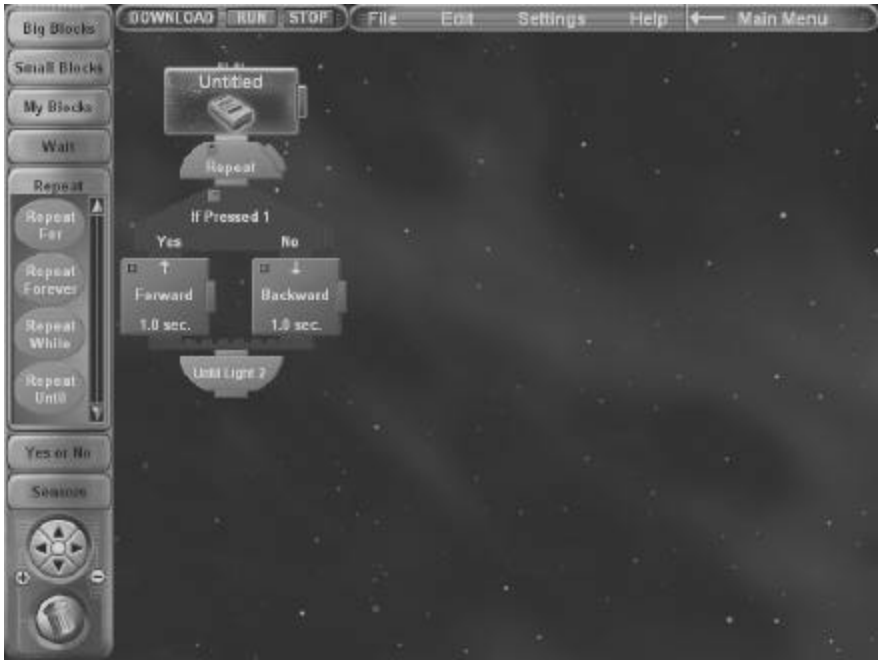

Figure 1. Mindstorms Programming Environment with Repeat and Yes-No Structures

Programs are downloaded to a programming block, shown in Figure 2, which allows for control of external devices and for input from sensors connected to the programming block. In addition to the programming constructs mentioned above, the system allows for the development of daemons, called listeners, to be developed to respond to external events. The listeners are subprograms that respond to conditions reported by the sensors. Using this, students are introduced to the concept of event programming that in turn is reinforced in latter portions of the class.

Figure 2. Programming Block with Sensors and Motors Connected

Student projects that require the program block to respond to specified inputs and to complete a specified task within a series of design constraints are developed. Three of these projects are completed during the first half of the course and a formal design report is required for each project. The first project required that the system move over a prescribed course is the shortest time possible. The second project required that the system move within a prescribed area to a goal utilizing the sensor system to locate the boundaries of the system and to locate the goal in the shortest time possible. The final project required that the system locate a goal within a maze littered with physical obstacles in the shortest time possible with a constraint on the number of parts available which could be used.

Each project requires that the design teams consider the problem from two viewpoints. First they had to construct a physical device from the parts supplied that could traverse the problem area and then they had to develop a control program that would respond to conditions and guide the system to the goal state. These two concerns were linked because selection of different types of physical devices changed the way in which the final construct could be controlled.

After all three of the Mindstorms projects were completed, the students began their study of Visual Basic as a programming tool. The methodology for teaching Visual Basic was reduced to teaching the syntax of the programming language while referring to the logical development from the Mindstorms portion of the class to emphasize the logical tools necessary for program development.

Results

The programming class has always been approached by the students with a skeptical attitude and with some trepidation. The typical attitude was that this was a course that was difficult and confusing and a course that was to just be completed and moved past. Anecdotes were passed down from upper class students about impossible, to them, assignments with no relevance to anything else that was being done in the curriculum. Teachers who attempted to introduce relevant content were faced with the problem of teaching the civil engineering content, the logical development of a solution, and the syntax of the language. Difficulty on the student's part in any of these three areas would generate a negative attitude that was very difficult to overcome.

In addition to these problems, or possibly because of them, the students would not generate the effort necessary to understand the process of program development. It was common for students to begin syntax development before any logical thought went into the process. Although an emphasis on a logical analysis and testing was made through the use of flowcharts, the students considered this a waste of time and so took an inordinate amount of time in developing their programs. The students, after developing their code, would

not utilize test cases to evaluate their programs. This lead to the submission of programs that might work for very limited cases but would have no general usefulness.

The Mindstorms added class entered with the same attitude as previous classes. They shared stories from other students about how hard this class was and about how much time was required for the class. One-fourth of the class had some programming experience from either high school or at the junior college level. With the exception of one student, all of the students who had prior experience did not consider that they could complete a programming task if asked to do so. A comparison of the attitudes of the students entering and exiting the class toward the difficulty of the material is presented in Figure 3.

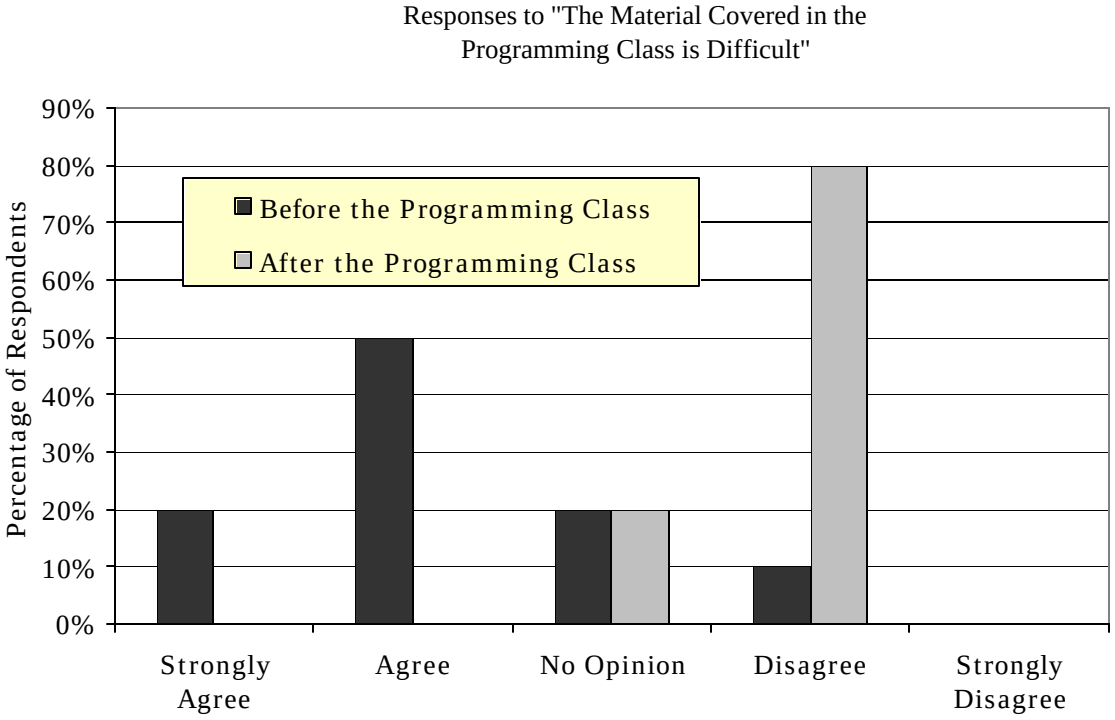

Figure 3 – Attitudes of Students Towards the Programming Class

When it was explained that for the first half of the class they would be working on design projects in the same manner that they had in their first two classes in the sequence, they grumbled about the writing but seemed relieved that they would not have any "impossible" programming assignments early. All of the students had exposure to some types of Legos so the construction part of the Mindstorms system was familiar to the students. In addition, the idea of Legos seemed to make the students feel comfortable with the learning environment. An unexpected consequence to the approach is that the students believe that this course "fits" into the sequence and into the civil engineering curriculum rather than as an unconnected part of the curriculum.

The programming environment was not familiar to the students but the relative ease in which simple programs could be developed within a very short period of time made the introduction into this environment also easier. With very little introduction, the students developed a simple system for guiding a programmed vehicle of their construction through a series or maneuvers during the first three-hour lab period. Working in small groups, all of the groups were successful and confident that they could proceed onto more difficult projects.

Lecture periods were devoted to looking at the logic of programs and how the simple syntax of the Mindstorms programming environment allowed for the physical problems presented to be addressed. One of

the most interesting points from the use of the Mindstorms was the ease with which the students accepted the idea of flow diagrams or flow charts. The visual Mindstorms programming environment presents programs as a series of blocks interlocked so that the program flows through the blocks. The Repeat structure and the If-Then-Else structure both have the one-way in and one-way out structures that are also emphasized in structured programming. The analogy between the program developed in Mindstorms and the flow chart for the If-Then-Else is illustrated in Figure 4.

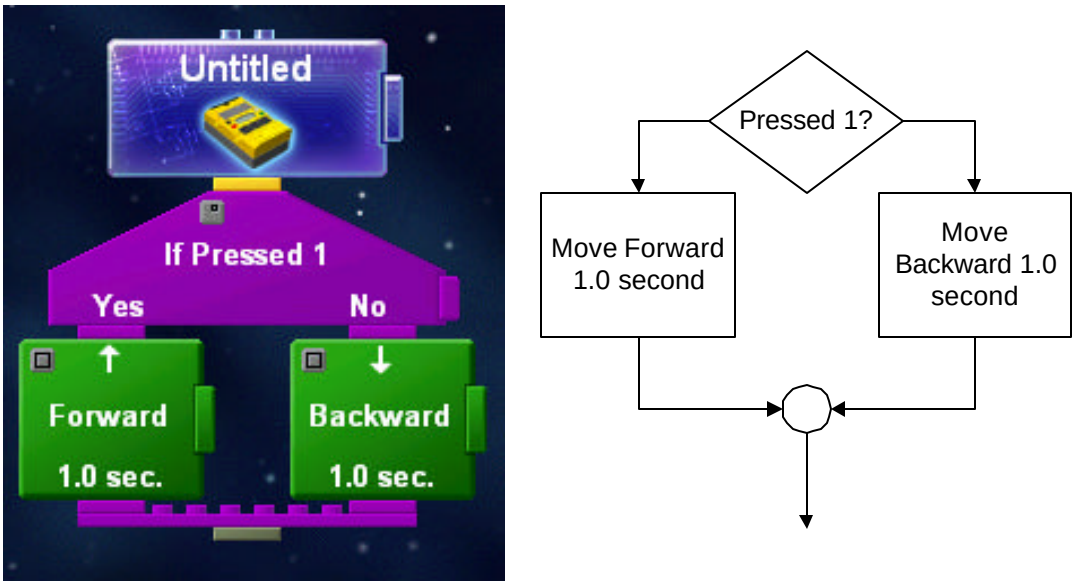

Figure 4. Comparison Between Mindstorms If-Then-Else Construct and Flow Diagram of Same Construct

The students began to think in terms of the blocks and how the logic/program would flow during the program run. When the shift came to using Visual Basic rather than Mindstorms, the students were used to developing this visual tool to see how the program executed. This was emphasized in all the problems that were used during the lecture as well as all homework problems used in Visual Basic.

With the separation of the logical and syntactical development, it was noted that the type of questions that students asked shifted from questions that blurred the line between syntax and logic to questions that focused on one of the areas. In an exit survey given to the students, eighty percent of the students felt comfortable with their ability to write a logical outline of a problem that could be converted into a Visual Basic program.

The students had an overall positive response to the new approach to teaching programming. Students with a previous unsuccessful programming experience noted that the shifting of logical development as a precursor to syntactical development made both of the concepts easier to use. In the exit survey, every student responding to the survey when asked if they would continue to use the Mindstorms as a teaching aid in this class responded positively. When asked for some of the reasons why Mindstorms should continue to be used, some typical student responses were:

- ...made the transition into programming much easier..
- ...helped you to understand the logic and process for programming..
- ...a fun and interesting way to learn programming..
- ...gives the students a basis and a working model of what programming is..

In addition to the exit survey, a comparison was made on the performance on similar programming assignments in Visual Basic from the previous two times that the class had been taught. An average improvement of 12% was made on similar assignments when they were preceded with the Mindstorms introduction. There was an improvement in both the logical work and a reduction of syntactical errors in the

works compared. While students in previous courses had been unable at times to even generate enough work for evaluation, this was not the case in this semester.

Conclusions and Future Plans

The overall positive responses from both the faculty involved in teaching this class and the students have reinforced the changes that have been made in the class. With the addition of a fourth class into the introductory sequence, there will be an additional course in which to reinforce programming concepts. In the spring of 2003, the fourth course will be used to develop simulations of engineering problems using Visual Basic to reinforce the work that the students have done in their introduction to programming. In addition, programming will now be considered a required skill for students entering the upper-division courses in Civil Engineering and an effort will be made to have faculty utilize programming more broadly and intensely throughout that portion of the curriculum.

References

Dembo, M.H. (2001) "Learning to Teach is Not Enough: Future Teachers also Need Learn How to Learn" *Teacher Education Quarterly*, V28 (4), 23-35

Heo, H. (2000) "Theoretical Underpinnings for Structuring the Classroom as Self-Regulated Learning Environment" *Educational Technology International*, V2(1), 31-51

Muller, J. (1998) "The Well-Tempered Learner: Self-Regulation, Pedagogical Model and Teacher Education Policy" *Comparative Education*, V34 (2), 177-193

Palazolo, P., Phillips, A., Camp, C. (2001) "Toys, Tinkerers, and Tomorrow: Growing Engineers," *Proceedings of the 2001 ASEE Annual Conference*

Perry, N.E., VandeKamp, K.O., Mercer, L.K., & Nordby, C.J. (2002) "Investigating Teacher-Student Interactions that Foster Self-Regulated Learning" *Educational Psychologist*, 37(1), 5-15

Piaget, J. (1972). *Psychology and Epistemology: Towards a Theory of Knowledge.* Harmondsworth: Penguin

Vygotsky, L. (1986). *Thought and Language.* Transl. And ed. A. Kozulin. Cambridge, MA: The MIT Press. (Originally published in Russian in 1934)

Zimmerman, B.J. (2000) "Self-Efficacy: An Essential Motive to Learn" *Contemporary Educational Psychology*, 25(1), 86-91

[1]*Paul Palazolo*

holds a Ph.D. in Environmental Engineering from The Georgia Institute of Technology and a MSCE and BSCE in Civil Engineering from Memphis State University. He is currently a faculty member in the department of Civil Engineering at the University of Memphis and the Assistant Dean of the Herff College of Engineering at the University of Memphis. He is part of a four-member team that developed and teaches the Foundation sequence of courses in the Civil Engineering department. He is also the director of the Joy of Engineering summer enrichment program for middle school students and teachers in the Memphis area where they are exposed to engineering problem solving and the engineering profession.

[2]*Anna Phillips-Lambert*

is Director of Technical Communications for the Herff College of Engineering at The University of Memphis in Memphis, Tennessee. She is also an active instructor and researcher in the Civil Engineering Foundation sequence. She received her B.A. degree in English from Memphis State University, her M.A. in English from The University of Memphis, and is completing doctoral studies in Counseling, Educational Psychology and Research at The University of Memphis.