12-2012

Effective Computer Programming Instruction for pre-University Albanian Students

Robert McCloud
Sacred Heart University, mccloudr@sacredheart.edu

Ardiana Sula
Sacred Heart University, sulaa@sacredheart.edu

Effective Computer Programming Instruction for pre-University Albanian Students

Prof. Dr. Robert McCloud[1], Ardiana Sula[2]

Abstract

The relationship between pre-university students and technology is frequently overrated. While we receive glowing reports about how young people are knowledgeable about computers, the truth is that their knowledge is typically about computer content and the manipulation of applications. Young students too often treat the actual programming and understanding of computers as a sort of magical mystery.

In this paper we look at a new Albanian initiative to identify and nurture the most talented of our pre-university students. In particular we look at contributions to the goal of making Albanians the most talented programmers in this area of Europe.

The study addresses the issue of how to get young people interested in programming by focusing on non-syntactic procedures. The authors consider two programming environments that teach through the use of objects and problem solving skills.

Albanian ICT education

In 2011 the authors worked together to produce the Final Report and Recommendations for Albanian ICT education through 2015. Describing the curriculum for grade 3, they wrote, "In this introductory year students will learn to relate to the computer as a machine that can enhance the quality of their life. They should understand both the power and dangers of computing. Students will also develop essential skills for care and regular maintenance."[3]

This report laid the foundation for a time to begin Albanian computer education: in the third grade.Later that same report addressed other issues such as Internet access, computer equipment, communication between classroom teachers and ICT specialists, and teacher training.

Although efforts are often limited by available funds, Albania has started to address some of these issues.[4]

1. *Prof Dr Robert McCloud, Sacred Heart University*
2. *Ardiana Sula, M.S., Ministry of Education & Sciences, Kosovo*
3. *Final Report and Recommendations, Ministria e Arsimit dhe Schenkes, Tirane, Albania, Prof. Dr. Robert McCloud*
4. *Integration of Information and Communication Technologies in Education, Ministeria e Arsimit dhe Schenkes, 2012*

There are 1,496 computer laboratories in the nation's public schools. In addition, each school has a dedicated broadband connection with speeds of 1024/256 Kbps.

New educational technologies to assist teaching have been installed in both Vocational Education and Training schools specializing in ICT curricula and 20 general high schools. The most pervasive technology is the Smart Board.

In Albania 4,484 pre-university teachers have now received ICT training. For the most part this training consists in ICT classroom integration and digital content development. To monitor and control ICT usage, about 500 school administrators are now trained to use module focusing on introduction, integration and support of national ICT policies. Most have not formal computer programming education.

Programming at young age

We like to think of all our young people as "digital natives". We watch as they rapidly input text messages with their thumbs, as they learn any new mobile application with ease. In short it often appears as if they were born with a digital facility that is not so native to those over 35. While admiring parents and relatives often point to these digital skills as a sign of intelligence, the concept of "born digital" in fact refers to facility, not to intelligence at understanding concepts.

While our young people excel at using technology, they are not so good at the creative aspects of this discipline. They can upload photos and posts, *re*-blog other people's content, do quick searches that lead to browsing, and display admirable focus in solving adventure game puzzles. If it involves drag-and-drop or following pre-created templates, they are natively at home. Should we call on them to actually write the program behind any of their applications, they are clueless. Computer programming is regarded as a dry, abstruse subject best relegated to geeks. Almost no one refers it to a creative activity.

From a United States perspective there was an initial enthusiasm that followed personal computer introduction in the late 1970s and early 1980s. Educators imagined a generation of young people that would embrace programming. As a discipline, computer programming was seen as a way to harness the tremendous potential inherent in this emerging technology.

In many ways the electronic spreadsheet epitomized computer programming potential for everyman. Dan Bricklin and Bob Frankston, two Harvard Business School students, watched as their professors laboriously used paper and blackboard to make financial calculations using spread-sheets. Whenever one assumption changed, the entire spreadsheet would have to be recalculated by hand.

The two young students thought there had to be a better way. Retreating to a Boston garage, they developed VisiCalc, the first electronic spreadsheet. We know VisiCalc today in its pervasively popular form, Excel.

When it was first distributed in 1979, VisiCalc almost singlehandedly made the personal computer possible. Before its introduction corporate executives were unlikely to have a personal computer. Since its main application was for word processing, it rapidly became regarded as a fancy electronic typewriter best suited for secretaries. Just as no self-respecting executive would type his own letters, so he would also not have a personal computer seen in his office.

Bricklin and Frankston's electronic spreadsheet changed all that. Suddenly the executives could do financial what-if calculations in their office. Since they view financial

management and strategy as their true job, it because fashionable to actually have your own computer in your office.

Orders from corporate America went out for personal computers. Convinced that the product demand was there, com-puter giants like IBM devoted significant resources to personal computer development and marketing. A new industry was born.

In fact these developments came about because of two students who under-stood programming and its potential for helping with everyday tasks. They were motivated by a sense of intellectual curiosity to creatively solve a problem. It is precisely this drive and curiosity that is missing for the huge majority of today's "digital natives". What happened? Resnick et al. point to three factors that dimmed the initial enthusiasm for teaching students to program.[5]

Students often stumbled on the syntactic aspects of early programming languages. Frustrated by attempts to correct errors caused by misplaced commas or capitalization mistakes, students would simply give up.

When the teacher would introduce programming, it was accompanied by activities like generating lists of prime numbers or creating simple line drawings. Students had trouble connecting these activities to daily experiences. Teachers themselves were part of the problem. Frequently they were incapable of providing help when things went wrong or to provide guidance and encouragement when a student met with initial success.

Noting that there were some worthwhile efforts to develop languages aimed at younger programmers, the creators of Scratch wrote that, "…we felt that it was important to make the floor even lower and the walls even wider – but still support the development of compu-tational thinking."[6] The "lower floor" would appeal to students as young as eight years-old. Wider walls would mean making computer programming be seen as a mode of thinking and problem solving.

In particular this might address the lack of gender balance often found in computer programming. Although the problem is more serious in the United States than it is in Albania, programming is increasingly seen as a male domain. By extending the breadth of its appeal, we can tap a much greater talent base.

Caitlin Kelleher also addressed the gender balance question. In her doctoral dissertation she wrote, "Numerous studies have found that girls begin to turn away from math and science related disciplines, including computer science, during middle school. By the end of eighth grade, twice as many boys as girls are interested in pursuing science, engineering, or technology based careers."[7]

Kellerher's dissertation advisor was Randy Pausch, the Carnegie-Mellon professor who led the development of Alice, the programming environment we will propose as the second part to our Albanian computer science program-ming education solution.

Although Alice was originally intended to increase the computer programming interest of middle school girls, the power of Carnegie-Mellon has pushed its evolution to an environment that extends teaching capabilities from middle school through the university level. At many universities (including that of the author) there is some discussion as to whether it is better to take an objects-first approach to introductory programming courses.

---

5. Scratch: Programming for Everyone, Resnick, Mitchel et al. Accepted for publication in communications of the ACM (CASM).

6. Ibid, p. 4

7. "Motivating Programming: using storytelling to make computer programming attractive to middle school girls", Kelleher, Caitlin, School of Computer Science, Carnegie Mellon University, 2006.

Alice's evolution has also confirmed the feeling that it is neces-sary to have teachers who have been thoroughly trained in the software. To this end there are annual conferences devoted exclusively to Alice pedagogy.

Teacher training

Teacher training is a topic that appears regularly in the research. There is no way to ignore the fact that computer programming is a demanding intellectual activity. However, the fact that it is demanding should not discourage us from making a plan.

In March and May of this year and in July of 2011 the authors met with groups of Albanian teachers. Topics discussed ranged from implementation of ICT into the curriculum to identification of gifted students.

During these meetings teachers expressed a desire for more training, but also recognized practical necessities caused by the economic situation. Unfortunately we cannot control economics. What we can do, however, is to make decisions that utilize the *available* resources in the most effective manner. Our challenge is to define "most effective".

We need to carefully examine all options with an eye toward results that provide the greatest return for precious expenditures. We also need to stay away from solutions that appear cheap and easy, but do not really work.

This appears to be the case with cascade learning. Theoretically cascade learning provides an effective methodology that enables teachers to share with each other. A small group of teachers receives training funds. They go off to conferences where they learn new technologies. Then they return to their districts to share training with other teachers, who share with still others, etc. The new learning cascades down through the system. While beautifully simple and effective in theory, cascade learning is overrated.

A World Bank study[8] criticized the cascade learning as practiced in Albania. It found that cascade learning often behaves like gossip. By this the study meant that the learning message often because diluted and/or distorted as it traveled from teacher to teacher.

The study also found that learning needed corrected and guided implementation. Without that practice, only teachers' verbal descriptions changed. Their actual pedagogical practice remained essentially unchanged.

In a way this is a bitter pill to swallow. We would like to think that it is sufficient to pick out leading teachers and invest in their advanced training. Following that training we would ask them to go back to their districts and spread the word.

That is not the way it worked out. When it got spread, the word was distorted or diluted. Cascade learning sounds like one of those ideas that it terrifically appealing in theory. When, however, that theory is actually implemented in the real world, it does not work. Or worse, it works part way. This "part way" gives hope for correction and prevents us from moving on.

Strategy for building programming enthusiasm in our schools

We have already seen how pre-university computer programming instruction is not perceived as a growing field in the United States. In Albania the situation is somewhat different.

One encouraging aspect for Albanians is that girls do not shy away from programming to the extent that their American counterparts do. Perhaps this is because of the

8. World Bank (1994) Strategy note.- Albania education system. As quoted in Leach, J. (1996) Learning in practice: support for professional development. In R. Mills, and A. Taid (eds).), Supporting the learner in open and distance learning (pp. 101-126). London, UK: Pitman Publishing.

traditionally strong Albanian tendency toward mathematics. Or perhaps it is because ICT is viewed as a field full of employment opportunities. Whatever the reason, this situation is encouraging.

Both the raw student ability and the tradition of mathematical logic are in place. What is needed is a national strategy for finding and nurturing talent.

We believe that this strategy is best implemented by building on current programming pedagogical wisdom: start programmers young; treat programming as a non-syntactic problem-solving exercise; make programming fun; have young programmers interact with each other; cultivate a teaching community that is passionate and willing to share.

In Albania our strategy should also recognize the fact that financial resources are limited. We need to find a solution that can be implemented locally, has no or low licensing fees, is relatively easy to learn, and has a large user community that is ready to help others.

We also believe that programming education should begin at a young age. This is a belief that was reinforced during our meetings with Albanian teachers. Those in attendance all agreed that the capability was there at a young age and also that they could identify those students who would be classified as "most talented". Their enthusiasm and opinions are supported by academic research. Douglas Clements and Dominic Zullo studied the effects of LOGO computer programming on six year-olds. They found that learning programming could help with "reflectivity, divergent thinking, meta-cognitive ability and an ability to describe directions."[9] It is not within the scope of this paper to define the exact age at which young people can effectively learn program-ming concepts. While Clements and Zullo concluded that the age is six, researchers at Massachusetts Institute of Technology feel the appropriate age is eight.[10] In Albania we feel the age should be set at eight for a practical reason: this is the minimum age recommended for learning the programming language called "Scratch".

Beginning at age eight

One has only to look at the Scratch homepage to see how this learning community is full of energy. The software is easy to download. Tutorials are readily available. Examples of Scratch users' work are plentiful. There is a large community ready to share knowledge. There is a design studio to teach users how to created digital art. And remixing of others' work is encouraged. After spending a few minutes looking at this homepage (Figure 2), one can see the learning and collaboration possibilities inherent in this platform. Its inviting nature asks you to participate through doing. As evidence of the program's effectiveness, one is offered simply the work of others. One of the most impressive tutorial features is that they are often narrated by very young voices. The Scratch group at MIT has assembled a array of students who want to help each other learn. Learning collaborations skills is itself a vibrant lesson to teach our young people.

When one looks at the homepage, another reminder fact strikes us: the language of programming and the Internet tends to be overwhelming English. The Scratch user community has already addressed this situation.

9. "Effects of Computer Programming on Young Children's Cognition", Clements, Douglas H. and Gullo, Dominic F., Journal of Educational Psychology, v 76, n6, p1051 – 58, Dec, 1984.

10. Resnick et al, op. cit., p 1

Figure 1: The Scratch website is available in these languages.

Albanian does not appear on the list. At this particular moment there is no possibility of translating into Albanian. However, that situation will change later this year. Translation has been suspended until the introduction of Scratch 2.0. As of late November, the new version has not appeared, but it is expected before the end of 2012.

Figure 2: The Scratch Homepage (http://scratch.mit.edu)

The authors have written to the Scratch project at MIT, inquiring about permission to translate documentation into Albanian. We feel such a translation project is particularly important because many young Albanians will not be proficient in English. If they do have a second language, they would likely be able to locate it among those already included in the availability list. Clearly, though, it would be preferable to have documentation available in a student's native language.

Other than the translation issue, Scratch is an ideal platform for young Albanians to learn programming. It is provided free. It is ongoing support by one of the top technical universities in the world. There is a vibrant user community. Teacher training is available. And the program works. Mindful of the success of robotics kits such as LEGO MINDSTORMS, the Scratch development team worked with a Lego metaphor. Give students a bunch of blocks, and they will start tinkering with them.

Figure 3: Sample Scratch scripts[11]

Describing the program's look and feel, they wrote, "Control structures like **forever** and **repeat** are C-shaped to suggest that blocks should be placed inside – and to indicate scoping. Blocks that output values are shaped according to the types of values they return: ovals for numbers and hexagons for Booleans…"[12]

By coordinating graphics and function, and by adopting a building block approach, Scratch teaches concepts through use, not through syntactic development. Students thus learn that programming can be a creative problem solving tool, not a linguistic exercise.

Why learn with Scratch?

The developers describe their goals, saying they want Scratchers to learn "important mathematical and compu-tational concepts, while also learning to think creatively, reason systematically, and work collaboratively."

Continuing, they add that their primary goal is "…to nurture the development of a new generation of creative, systematic thinkers who are comfortable using programming to express their ideas." If the developers are correct, adopting Scratch for young Albanian students would have implications far beyond computer programming. We would be training a generation of structure creative thinkers.

First recommendation

Analyzing their website, Scratch devel-opers found that the main age range was between eight and 16. They found that usage peaked at age twelve.

Our recommendation is to begin a pilot project with users as young as eight years-old. If students in other countries can start this early, there is no reason Albanian students cannot also follow suit. Scratch is so intuitive, and the tutorials (made by peers) are so helpful, that there is no good reason Albanian students cannot begin their program-ming careers at this age.

At this point we can only guess at what would be the best age to switch from Scratch to a more advanced pro-gramming environment. We will make that guess and say it should be about twelve or thirteen.

Part of that guess is based on current experience. If Scratch web usage peaks at twelve, there must be a reason for that peak. Our possible reason might be a loss of interest in programming. However, we suspect that the real reason is that students move on to a more demanding programming environment.

Some might be ready to transition to a procedural object-oriented language as then enter high school. This, however, would almost certainly be a small min-ority. Even though 12 – 13 year-olds are thought to have the analytic skills to follow abstract programming logic[13], current pedagogy does not typically include them in syntactic programming lan-

11. ibid, p. 4

12. ibid

13. "A Theory of the Relationships between Cognitive Requirements of Computer Programming Languages and Programmers' Cognitive Characteristics", by

guage instruction either in the United States or Albania.

A more fruitful course would be to find another programming environment that would take advantage of cognitive development by introducing object-oriented concepts in a non-syntactic environment. This leads us to the next step.

Second recommendation

There is another programming development environment that requires creative, systemic thinking without introducing strict linguistic syntax. It is Alice, developed by Carnegie-Mellon University.

Alice is an interactive, animation based programming environment designed to teach through manipulation of objects. There is a substantial library of characters available for animation. Alice users are also encouraged to develop new objects that are added to an always growing library. In classroom use Alice users focus on creating animated stories or computer games.

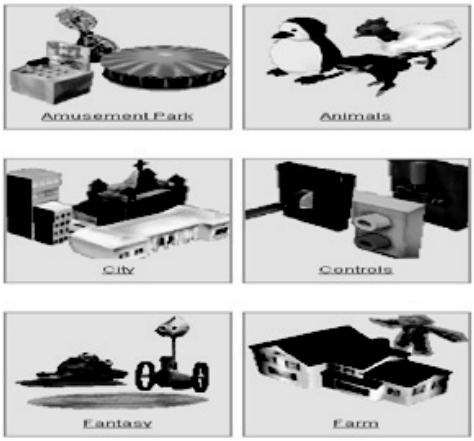

Figure 4: Samples from the object gallery (http://www.alice.org/index.php?page=gallery/index).

A particular strength of this programming environment is its ability to use 3D graphics. Each of the objects above links to an additional subject library.

Although students can create objects on their own, a typical learning curve begins with animating those from the existing libraries. There is also an extensive sound library available. Sound categories include: background music, musical cues, and a long list of sound effects such as alarm, beast, cat, dragon, footsteps and shark. There is also help available for creating your own sounds.

Extensive support available

Alice is primarily an English-based environment, although there is extensive support in Spanish. Carnegie-Mellon is beginning a translation project. Again, the authors have written to Alice developers asking if we can develop an Albanian version of the platform.

Like Scratch, Alice is offered as free. Carnegie-Mellon supports the software and also maintains an active listserv where users post questions, suggestions, and assist each other. This year the Third Alice Symposium will be offered at Duke University in Durham, North Carolina. For the first time there will be workshops in Bogota, Columbia, sponsored by the Columbian Ministries of Technology and Information and Education.

If we had a national commitment to using Alice in Albania, there would be an opportunity to eventually host such a workshop for southeast Europe, a step toward establishing this country as a force in computer programming education. The latest Alice version, 3.1, has been designed to serve as a transition to Java. If we were to utilize Alice in

our curriculum, it could ultimately be part of a three-step programming curriculum from third grade through high school graduation, Students would begin with Scratch, transition to Alice in middle school, and begin learning Java in their last two high school years.

Beginning the program

A wonderful aspect of this proposal is the fact that it does not involve expenditures for capital goods or software. Scratch and Alice are both completely free.

Our recommendation is that whatever monies are available be devoted to teacher training. If a program like depends on cascade learning, it stands a good chance of failure. Rather a strong program for finding and nurturing teachers is necessary.

There is already a proposal in place to offer Scratch and Alice courses for teachers here in Albania during the summer of 2013. If the proposal is funded, it would be possible to introduce these programming languages to the curriculum in fall, 2013.

In addition, Johns Hopkins University's Center for Talented Youth has offered us five full scholarships for Albanian students to go to the United States in summer 2013. This offer is dependent on our raising funds for at least five additional students.

We could also send teachers to the Duke symposium and to summer teaching courses offered by Johns Hopkins. The Center for Talented Youth also offers an online course for Scratch programming. Duke is stronger in its Alice offerings.

There is a tremendous opportunity available to implement a low cost program that would put Albania on a path toward leadership in computer programming. Such a path would also take advantage of the efforts already made to upgrade the hardware and networking capabilities in our national school system.

It is time to seize that opportunity.