Could you help me to change the variables? Comparing instruction to encouragement for teaching programming

Dimosthenis Makris
Ionian University
Corfu, Greece

Kleomenis Euaggelopoulos
Ionian University
Corfu, Greece

Konstantinos Chorianopoulos
Ionian University
Corfu, Greece
choko@ionio.gr

Michail Giannakos
NTNU
Trondheim, Norway
michailg@idi.ntnu.no

ABSTRACT

Computer programming has become an important skill and it can be taught from early school years. Previous research has developed and evaluated several visual programming tools that are suitable for computer education in schools. However, little is known about how pedagogic styles affect student attitudes towards learning computer programming. This paper reports on a preliminary study on the influence of alternative teaching styles on student's enjoyment and attitude towards computing. Two groups of twelve students each were asked to revise a computer game. The traditional instruction group was provided with detailed information, while the encouragement group was asked to help the teacher to change the variables of the game. The results indicate that an encouraging pedagogic style promotes more positive attitudes towards computer programming and more self-confidence than traditional instruction. Further research should repeat the experiment across several weeks for more programming concepts and should also assess the cognitive benefits.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer Science Education, Curriculum.

General Terms

Measurement, Experimentation, Human Factors.

Keywords

Programming, Secondary education, Computational thinking, Computer education, Scratch, encouragement, confidence

1. INTRODUCTION

For many years computing has been included in the curriculum as an important discipline in secondary education. Moreover, there have been several approaches to improve programming literacy in schools, mostly in terms of visual programming tools. Nevertheless, recent reports by the Computer Science Teachers Association [20], and the ITiCSE Working Group: Informatics in Secondary Education [8] revealed that computing courses face problems regarding their lack of exposure and motivations. We suggest that in addition to visual programming tools, educators should also consider individual learning styles, as well as alternative pedagogic styles that are better suited to the unique aspects of computational thinking, such as the creative, tinkering, and making aspects [3] of the computing culture.

WiPSCE '13, November 11-13, 2013, Aarhus, Denmark.

Currently, there are multiple efforts to broaden participation in Computer Science and introduce computational literacy to young students [15][17]. Contemporary computer education has emphasized the consumption (i.e., the use) of technology instead of the creation and understanding of it [6]. For example, Resnick [14] has argued that, by learning to program, children are able to learn in a more meaningful way, which is more motivating and results in deeper and better learning. They made an analogy with pianos and stereos. Even though it is easier to play a stereo, playing the piano enables someone to create, to express personality, and to develop a deeper relationship with music.

Previous research suggests that visual programming tools provide a positive experience for first-time programmers. Researchers and educators have been developing visual programming environments that are customized to children, such as LOGO [16], Alice [4], Greenfoot [9] and Scratch [10]. Although there are several visual programming environments, there is limited research about the teaching practices that are most effective for involving students with programming.

In addition to research on tools for ICT education, some studies have analyzed the pedagogical style. For example, Wick [19] highlights the influence of the first day of lecture on early attitudes of computer science students. One of most interesting pedagogic styles is coming from outside of ICT education. Mitra and Dangwal [13] proposed to let children learn, on their own, about basic molecular biology in English (which is not their mother language) with the help of a friendly mediator, who had no knowledge of the subject. They found that children performance was comparable to children who learned the subject from qualified teachers. Their findings suggest that any topic from simple internet browsing and language learning, up to molecular biology might also be facilitated by a friendly and encouraging mediator. Notably, if the mediators have no knowledge of the subject matter, it prevents them from delivering the answer to the students.

The above experiments are part of broader philosophy towards pedagogy. Mitra [12] has emphasized: "we need real schools, not

factories that convert our children to standard machines. Most schools today are the product of an expired age; standardized curricula, outdated pedagogy, and cookie cutter assessments are relics of an earlier time. Schools still operate as if all knowledge is contained in books, and as if the salient points in books must be stored in each human brain -- to be used when needed. Students are rewarded for memorization, not imagination or resourcefulness." As a matter of fact, most schools around the world teach programming in the aforementioned fashion. Students learn the principles of computer programming similarly to History, without the freedom to create something on their own, without discovering that programming with accessible visual platforms like Scratch can be creative and amusing.

In our work, we are exploring whether the above encouraging pedagogic style is applicable to teaching programming in secondary education. The main role of an encouraging teacher is to motivate children to continue their learning exploration. From here on, we refer to this approach as the "Sugata Mitra" style of teaching. There have been interesting findings on visual programming tools and on computer education pedagogy, but these two streams of research have been independent. Overall, we suggest that the practice of making cannot be independent to the pedagogy of a making culture.

The research question that guided our study is: "How does the encouraging style of teaching affect children's attitude and self-confidence towards computer programming?

2. METHODOLOGY

We explored young students' attitude towards alternative teaching styles for computer programming at a lower secondary education school (ages 12-15) of Greece (Gymnasium). In particular, we are comparing the traditional instruction of programming to the encouragement style, which has been promoted by Mitra [12]. Our sample consisted of 24 students from a Gymnasium and we employed the Scratch visual programming tool [10]. Scratch is an educational programming platform that allows users to create interactive games, animations, presentations, arts, etc., simply by dragging and dropping command blocks. These blocks can be handled like pieces of a puzzle and they represent concepts such as variables, conditions, loops and boolean expressions. Children can create their own Scratch projects or build upon existing ones, becoming actual designers of games, animations, and stories. In this study we employed the openly available Alien Shooter game[1], which can be found in the official Scratch Gallery.

We conducted a between-groups experiment with two groups of 12 students each. The group assignment was done without a pre-test, because the purpose of the study is to measure attitude and not cognitive performance. The teacher was the same person for both groups, and he was knowledgeable about computer education in general but he was not knowledgeable about the particular study design and the respective pedagogic issues. Moreover, the teacher is independent from the researchers and he does not have any conflict of interest in favor or against one of the methods.

The first group was instructed about programming concepts in what we refer to as the "Traditional" style. That means that the teacher used the Scratch tool to demonstrate to the children how to change variables in the game (the score and the ship-health). The second group learned about the same programming concepts in an encouraging style. In this case, the teacher did not provide any instruction about the variables theory, but encouraged the students to change the score and the ship-health features of the game in Scratch, thus the students indirectly worked with the same variables concept (Figure 1).

In summary, the ship-health and the score variables were changed in two ways: 1) the teacher instructed the students how to change them by demonstration on the video projector, and 2) the students were encouraged to help the teacher change them by working individually on their computers (Figure 2).

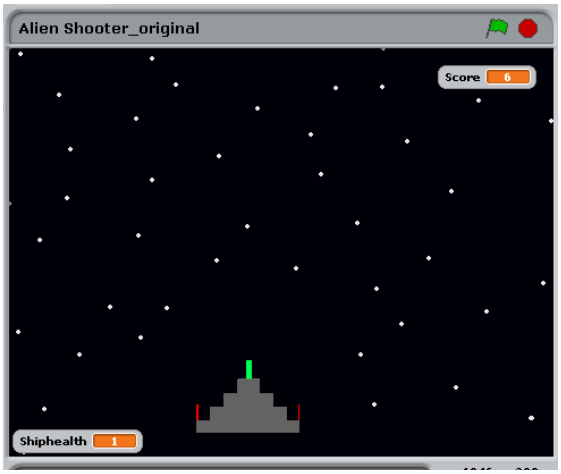

Figure 1 The score (top-right) and the ship-health (bottom-left) variables were changed in two ways: 1) the teacher instructed the students how to change them, and 2) the students were encouraged to help the teacher change them.

In order to measure children's attitude towards programming we adopted the questionnaire proposed by Giannakos et al. [6], which assesses performance expectancy, satisfaction, self-efficacy, social influence, perceived behavioral control and behavioral intention. We also used the Self-Assessment Manikin (SAM) by Bradley & Lang [1] in order to understand more about students' affective responses towards the activity. SAM is a pictographic questionnaire that assesses affective responses in three dimensions: valence, arousal, and control. The SAM instrument rates emotion on a scale from 0 to 9, across the two sides of an emotion: pleasure (sad - cheerful), arousal (quiet - active) and control (independent - dependent). Moreover, in order to measure students' knowledge on the subject, they were asked to answer questionnaires prior to the learning activities as well as after them.

Figure 2 The task was to change the increment of the variable

3. RESULTS

According to the results from the attitudes and the SAM questionnaires, the Sugata Mitra group assessed themselves as having acquired more knowledge about programming. Moreover, they appeared to have liked Scratch's environment and enjoyed the activity more than the Traditional group. Notably, on the statement "You will continue to study programming" the Traditional group scored 35% positive answers in contrast to the 80% positive answers of the Sugata Mitra group (Figure 3). Moreover, on the "You will study programming on a regular basis" question, the results are similar with 25% positive answers on Traditional group and 60% on Sugata Mitra.

[1] Alien Shooter game: http://scratch.mit.edu/projects/1785051/

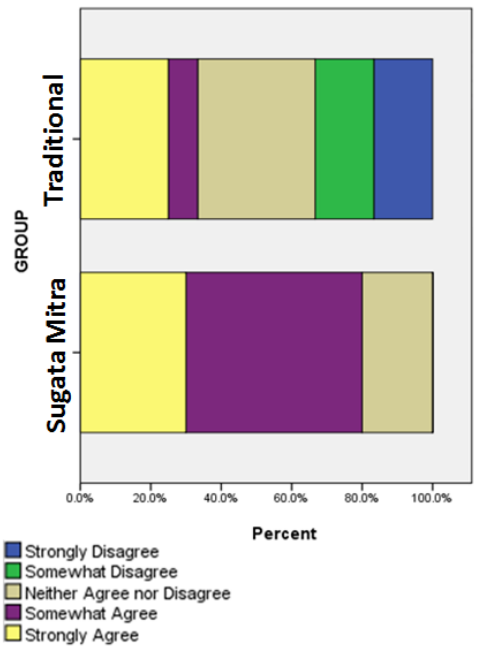

Figure 3 Results on "You will continue to study programming"

Regarding the use of Scratch in the classroom as an instrument for learning programming, the students of the Sugata Mitra group liked it more and want it to use it again. For the question "You plan to use Scratch again" only 50% of the Traditional group was positive, as opposed to Sugata Mitra's 100%. The same pattern is observed on the statement "You hope that you will continue using Scratch at school" with 55% positive answers of the Traditional group and 100% for Sugata Mitra.

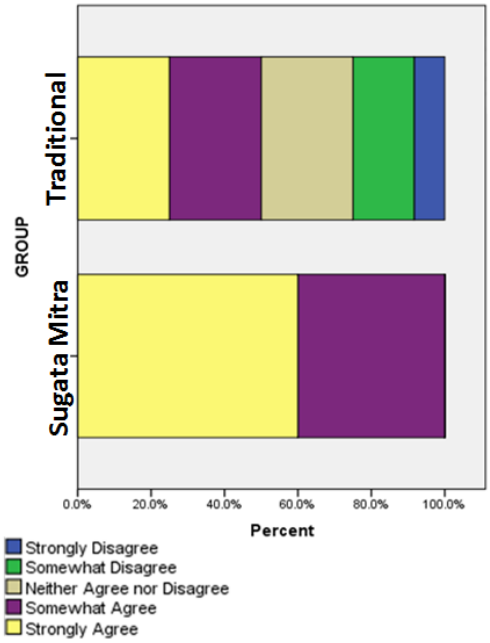

Figure 4 Results on "You plan to use Scratch again"

The results from SAM indicate that students of the Sugata Mitra group found the activity more enjoyable and arousing. They also considered themselves to be more independent, which is an indication of increased self-confidence (Figure 5). Moreover, before the experiment 90% of both groups were positive that "they know what programming is". However, following the activities, only 45% of the Traditional group were now positive, as opposed to Sugata Mitra's 100%.

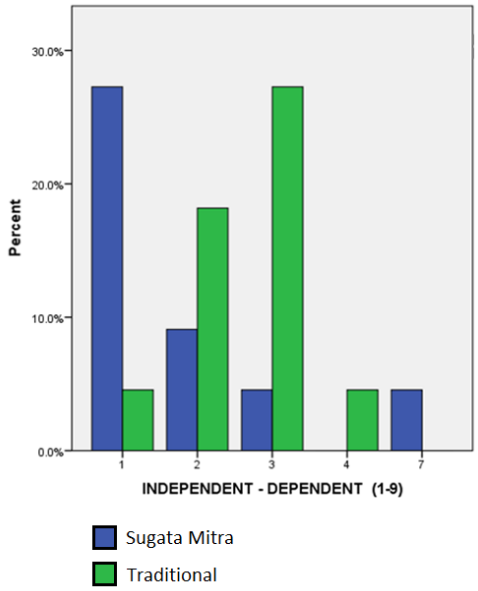

Figure 5 SAM: Measuring Control between groups

4. CONCLUSION AND DISCUSSION

The findings indicate that the teaching approach influences the students' attitude towards the course of computer programming. In our study, it seems that students in the traditional instruction group are less likely to get involved with programming in the future than the participants in the encouragement group, because their experience was less stimulating. Our findings suggest that once the students are given an opportunity to be creative and unique, as well as feel good about their ability to figure things out on their own, their interest in computer programming increases.

Moreover, the results of the affective response indicate that the encouragement group was more excited and felt more in control of the activity. It is worth pointing out that it is unclear whether the encouragement method might also result in better programming performance, as measured by computational thinking tests. However, our study provides evidence that the encouragement approach to programming with a visual tool results in a greater intention for future involvement with programming which the authors feel that is more important in bringing new people in the field of information technology.

The findings of this study should be interpreted in the light of some limitations. The study was based mostly on short-term quantitative data collection from a small sample size, which entails certain disadvantages on the generalization of the findings. Moreover, we have performed an experiment that is based on a very simple concept: changing the variables that hold the score and ship-health for a video-game. Computational thinking is much more than variables [2] and computer programming tools like Scratch can be also used for creating animations and stories, so it has to be evaluated for those concepts too. Moreover, the Alien Shooter game is a Space Invaders clone, so it might not be suitable for some of the individual learning styles.

Although students use computers for many tasks both at home and at school, the majority of them fail to comprehend what computer science is and how it relates to computational thinking and problem solving. Their exposure to computers in school usually consists of word processors and media presentation tools. Few secondary education schools have a mandatory (or even an elective) computer programming course during this crucial time where students are starting to think about career choices and making future educational and vocational decisions.

Despite these limitations, the proposed pedagogic method has generated valuable insights on the notion of encouragement for teaching programming in secondary education and has opened new avenues for conducting in-depth studies. In further research, we plan to repeat the same experiment across multiple weeks of ICT education with a bigger sample of students and to employ qualitative data collection methods, such as interviews. Finally, further research, should also measure the learning performance of the students. Although the students seem to increase self-confidence and liking of computer programming through the encouragement teaching style, it is unclear whether they actually improve their cognitive skills too.

5. Acknowledgements

We are grateful to the teachers and students for participating in the study and to XXXX for assisting the preparation of early drafts of this paper.

6. REFERENCES

[1] Bradley, M.M., Lang, P.J. 1994. Measuring Emotion: The Self Assessment Manikin and the Semantic Differential. *Journal of Behavioral Therapy and Experimental Psychiatry*, 25, 49-59.

[2] Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada.

[3] Chorianopoulos, K., Jaccheri, L., & Nossum, A. S. (2012). Creative and open software engineering practices and tools in maker community projects. In Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems (pp. 333-334). ACM.

[4] Cooper, S., Dann, W., Pausch, R. 2000. Alice: *a 3-D tool for introductory programming concepts*. *Journal of Computing Sciences in Colleges*, 15, 5, 107-116.

[5] Ding, W. and Resnik, M. 2013. http://www.ted.com/talks/mitch_resnick_let_s_teach_kids_to_code.html (Last accessed: Feb., 2013)

[6] Giannakos, M.N., Hubwieser, P. Chrisochoides, N. 2013. How Students Estimate the Effects of ICT and Programming Courses. In *Proceeding of the 44th ACM technical symposium on Computer science education.* SIGCSE '13. ACM, New York, NY, USA, 717-722

[7] Grover, S., Pea, R. 2013. Using a Discourse-Intensive Pedagogy and Android's App Inventor for Introducing Computational Concepts to Middle School Students. In *Proceeding of the 44th ACM technical symposium on Computer science education*. SIGCSE '13. ACM, New York, NY, USA, 723-728.

[8] Hubwieser, P, Armoni, M, Brinda, T, Dagiene, V, Diethelm, I, Giannakos, MN, Knobelsdorf, M, Magenheim, J, Mittermeir, R, and Schubert, S. 2011. Computer science/informatics in secondary education. In *Proc. of the 16th annual conference reports on Innovation and technology in computer science education - working group reports.* ITiCSE-WGR '11, ACM, NY, USA, 19-38.

[9] Kölling. M. 2010. The Greenfoot Programming Environment. *Trans. Comput. Educ*. 10, 4, Article 14 (Nov. 2010), 21 pages. DOI=10.1145/1868358.1868361

[10] Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E. 2010. *The Scratch Programming Language and Environment*. *Trans. Comput. Educ,* 10, 4, Article 16.

[11] Malan, D. J., Leitner, H. H. 2007. Scratch for Budding Computer Scientists. In *Proceedings of the 38th SIGCSE technical symposium on Computer science education.* SIGCSE '07. ACM, New York, NY, USA,223-227.

[12] Mitra, S. We need schools… not factories,. http://www.huffingtonpost.com/sugata-mitra/2013-ted-prize_b_2767598.html (Last accessed: May, 2013)

[13] Mitra, S., Dangwal, R. 2010. Limits to self-organising systems of learning—the Kalikuppam experiment, *British Journal of Educational Technology*, 41, 5, 672-688.

[14] Resnick, M., Bruckman, A., Martin, F. 1996. Pianos not stereos: creating computational construction kits. *Interactions*, 3, 5, 40-50.

[15] Soh, L. K., Samal, A., Scott, A., Ramsay, A., Moriyama, E., Meyer, G., Moore, B., Thomas, W. G. and Shell. D. F 2009. Renaissance computing: an initiative for promoting student participation in computing. SIGCSE Bull. 41, 1 (March 2009), 59-63

[16] Solomon, C. J. 1978. Teaching young children to program in a LOGO turtle computer culture. *ACM SIGCUE Outlook*. 12, 3, 20–29.

[17] Webb, D. C., Repenning, A and Koh. K. H. 2012. Toward an emergent theory of broadening participation in computer science education. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education.* SIGCSE '12. ACM, New York, NY, USA, 173-178.

[18] Weng, J.F., Kuo, H.L., Tseng, S.S. 2011. Interactive Storytelling for Elementary School Nature Science Education. In *Proc. of the 11th IEEE International Conference on Advanced Learning Technologies*. ICALT '11. 336-338.

[19] Wick, M.R. 2007. Bridging the Conceptual Gap: *Assessing the Impact on Student Attitudes toward Programming*. In *Proceeding of the 38th ACM technical symposium on Computer science education.* SIGCSE '07. ACM, New York, NY, USA, 509-513.

[20] Wilson, C., Sudol, L., Stephenson, C., Stehlik, M., 2010. Running on empty: the failure to teach K-12 Computer Science in the digital age. ACM, (Last accessed: June, 2013) http://www.acm.org/runningonempty//fullreport.pdf