Programming for Non-Programmers – Fostering Comprehension Capabilities by Employing a PRS

Rachel Or-Bach
The Max Stern Yezreel Valley College
Emek Yezreel 19300, Israel
orbach@yvc.ac.il

Abstract: The study reported in this paper involved the employment of specific in-class exercises using a Personal Response System (PRS). These exercises were designed with two goals: to enhance students' capabilities of tracing a given code and of explaining a given code in natural language with some abstraction. The paper presents evidence from the actual use of the PRS along with students' subjective impressions regarding both the use of the PRS and the special exercises. The conclusions from the findings are followed with a short discussion on benefits of PRS-based mental processing exercises for learning programming and beyond.

Keywords: Novice programmers, comprehension, tracing, personal response systems

1 Introduction

Throughout the short history of personal computers there have been attempts to delineate possible benefits of teaching programming to non-programmers, from children to higher education students. The potential benefits range from motivation and creativity to logical thinking, problem solving skills and more (diSessa, 2000; Kay, 1991; Papert, 1980). Teaching programming to students who have no special interest or inclination towards programming and related topics requires at least two things: providing students with some motivation and carefully designing the instructional materials and strategies. A course titled "Design of computerized games and interactive stories" was developed to address these requirements. A multimedia interactive development environment and a goal of designing a working game were chosen to enhance stu-

dents' motivation. Special emphasis in the instructional materials was given to the relationship between explaining, tracing and writing skills in introductory programming (Lister, Fidge, Teague, 2009). A previous paper (Or-Bach, 2013) dealt with the contribution of such a course to the development of higher order cognitive skills and ICT competencies. The study reported in the present paper involved the employment of specific in-class exercises using a Personal Response System (PRS). These exercises were designed to enhance students' ability to explain a given code in natural language with focus on abstracting the goal of the code. This paper presents some evidence from the PRS use along with students' subjective impressions regarding the use of the PRS and the special exercises.

2 Related Work

2.1 Learning to program – The role of comprehension exercises

A significant distinction in the literature (see for example Robins et al., 2003) is between studies that explore program *comprehension* and those that focus on program *generation*. No doubt that these two topics are correlated, as comprehension activities must be performed during development and debugging. Lister et al. (2009), in an effort to build and refine the SOLO taxonomy for programming learning, stress this issue and the use of assessment tasks of "explain in plain English" that require the ability to grasp the overall goal or meaning of the code. Studies they carried showed evidence of a relationship between explaining, tracing and writing skills in introductory programming. They claim with pedagogical implications that until students have acquired minimal competence in tracing and explaining, it may be counterproductive to have them write a great deal of code. Robins et al. (2003) identify the issue of mental models as an important factor in learning and teaching programming. They claim that writing a program involves maintaining different kinds of "mental model". They distinguish between the model of the program as was intended, and the model of the program as it actually is that requires the ability to trace code for predicting its behavior. In our study both skills of tracing and explaining played a central role.

2.2 Games, education and personal response systems

The use of computer games for learning is widely advocated. Computerized games can provide a context for developing various skills, but constructing

such games seems to entails even wider educational benefits. Robertson and Howells (2008) argue that authoring a game can engage students in authentic rich tasks offering a good degree of learner autonomy. Vos et al. (2011) report on a study where they compared using a game versus constructing a game. The results suggest that constructing a game might be a better way to enhance student motivation and deep learning than playing an existing game. Game design is becoming a popular strategy for enhancing students' interest and skills with computer technology, deepening understanding of scientific principles, fostering critical media literacy and more (Hayes et al, 2008). Personal Response Systems (PRS) are technologies that facilitate interactivity with an audience. Such systems enable participants to instantly respond to posed questions using some end user device. The system can receive participants' responses and can provide a representation (or several ones) of the collected data. Instructors in variety of disciplines are increasingly using audience response systems to increase participation, engagement and learning (See for example, Beatty, Gerace, 2009; Mareno et al., 2010). SMS-HIT (Kohen-Vax et al., 2012) is a personal response system based on mobile devices for SMS and web response provision. The system is designed for teaching purposes enabling instructors to prepare and enact personal response activities in actual instructional setting for any subject domain. Readymade activities are stored in a repository and could be later on copied, modified and reused. The SMS-HIT system was used in our study for several types of in-class activities, as will be described in the next section.

3 The Course – General characteristics of the course

The rationale for the "Design of computer-based games and interactive stories" course was to provide a motivating and engaging context for introducing computer programming to students in the behavioral sciences departments in our college. This elective course is already offered for several years and is accompanied by an ongoing action research. The instructional approach, intermediate assignments and research tools were refined during the years. The programming environment chosen for this course is Scratch, which is a visual programming environment that lets users create interactive, media-rich projects (Resnick et al., 2009; Maloney et al. 2010). A key goal of Scratch is to introduce programming to those with no previous programming experience. Programming is done by snapping together command blocks to control sprites. Specific blocks can be placed on top of a stack of blocks to trigger that stack

in response to some run-time event. Multiple stacks can run at the same time to show simultaneous acts by different sprites. The programmer can watch stacks in the scripting area high-lighted when the action unfolds on the stage thus showing how the constructed scripts are interpreted by the computer. The course final assignment is the implementation of a game or interactive story using Scratch. Classwork consists of examples that students explore by "reading" and/or by executing the program; as well as other examples where students have to modify or construct a program in order to produce prescribed outcomes. In the last semester a PRS was employed with specially designed in-class exercises dealing with tracing and explaining Scratch scripts.

4 The Study – Participants, setting and tools

During the semester when this study was conducted the course lasted 14 weeks with a 2 hours session each week in a computer lab and 23 students participated in the course. The SMS-HIT PRS was used in most of the sessions, sometimes more than once but with an attempt not to use it too often. The exercises were presented through our LMS and included a link to a site where the response can be written or selected. The goal of the study was to investigate the contribution of these specific exercises to the learning of Scratch programming. Two types of questions were used: short free text ("What will be seen on the screen after executing the following script?"), and multiple-choice ones ("Do the two following scripts act the same? (Yes/No)"). The responses and their relative frequency were presented to the class at the end of the activity for further discussion and/or follow-up learning activities. A survey was administered to the students at the end of the course to collect students' impressions from the use of the PRS. The survey included 10 Likert-type items and a free text question regarding the special in-class exercises. An additional item dealt with the student's evaluation of his/her mastery of Scratch with relation to the class. Two of the 10 items dealt with factors related to the use of PRS in general (motivation and engagement) and the rest dealt with specific factors of the PRS use that may contribute to the learning of Scratch programming (requirement of mental processing, presentation of different answers, requirement to compare between scripts, discussions based on students' answers etc.) The scale for the Likert-type items was designed to obtain a finer resolution on the positive side of the scale because a goal of the study was to distinguish between the contributions of the different factors and the instructor's impression was that students favored in general the PRS use. The resulted scale includ-

ed the following categories: Do not agree, Agree slightly, Agree moderately, Agree, Agree strongly.

5 Findings

The average response rate for the PRS based in-class exercises was 15 out of 23 course participants. The fact that not all the students provided responses for these exercises might be because the chosen PRS allows anonymity (anonymity might also encourage participation). For most of the multiple-choice questions, the frequency diagram that was produced and presented to the students exhibited a distribution of opinions. Looking at the class results, students were asked to re-examine their response by re-reading the code. Only then they were asked to actually check the behavior of the script(s). It seemed that for many students the combination of mental tracing along with the actual execution is required for the ultimate conviction. For exercises with free-text short responses, the resulted variety of responses enabled discussions that dealt with what answers are actually the same, what is correct, and what might be the cause for other responses. These discussions were productive for stressing the need for accuracy, for re-examining the various programming structures and for presenting possible misconceptions. Twenty students (out of 23) completed the survey. One item of the survey dealt with students' perception about their relative mastery of Scratch programming. It seemed important to check this because over-confidence or under-confidence might give a different interpretation for the survey results. The results show that in general students felt good about their mastery of Scratch programming. The averages for the different survey items were in the range 3.4–3.95, indicating appreciation for the various contributions of using the PRS. The standard deviation was around 1. The highest average (3.95) was for item 5 – "Mental execution of a script helps to deeply understand specific commands". The lowest average (3.4) was for item 3 – "The fact that I saw additional answers made me re-think my answer". None of the students chose "Do not agree" for items 1 and 2 that dealt with general contributions (motivation and engagement respectively). The averages and the distribution for the survey items are quite similar, not showing differences between students' perceived contributions for the various aspects of the PRS use. Thus further analysis was carried out with regard to the students. It turned out that the standard deviation for a student (across items) was around 0.5, while the standard deviation for an item was around 1. We also checked for each student the differences between the average appreciation of using a PRS (items 1 and 2) and the average appreciation for the specific use of a PRS for

learning programming (items 3–10). The differences ranged between -0.9 and 1.9, indicating that different students experienced the use of the PRS differently (or interpreted it differently).

6 Conclusions and Discussion

Investigating difficulties of non-programmers, as well as respective instructional strategies, can be informative also for CS educators. In this study we tried to deal with the tracing and explanation skills, building on the evidence that was found of the relationship between explaining, tracing and writing skills in introductory programming (Lister et al., 2009). The general impression from the course, in comparison to previous years, is that the extensive use of tracing and explanation exercises was productive for learning to program with Scratch. From the instructor point of view, the in-class PRS-based activities were helpful for discovering students' difficulties and addressing them more effectively in class. The motivation and engagement provided by the PRS cannot be separated from the experience. Students' final submissions indicate good mastery of programming in Scratch based on what was learned in class and some self-learning. Results from the survey showed that in general students felt good about their programming capabilities in relation to the class. Results show that students had positive attitudes towards the PRS use during the course. The item that dealt with the use of mental execution of scripts for better understanding had the highest average and the highest number of students choosing "agree strongly". This fits well with the goals for the PRS-based exercises design. The fact that the distribution for all the items is similar might be because the various factors are highly correlated and students cannot differentiate between them thus consolidating a general attitude. This might explain also the fact that the variance among the students was much larger than the variance among the survey items. The tracing and explanation exercises can be beneficial also to the development of higher order cognitive skills related to abstraction, reasoning and use of mental models. Exercising "mental processing" seems important for current students that use technology extensively and tend to solve problems by tinkering or by very short chains of reasoning.

References

Beatty, I. D., Gerace, W. J. (2009). Technology-enhanced formative assessment: A research-based pedagogy for teaching science with classroom response technology. *Journal of Science Education and Technology*, 18(2), 146–162.

diSessa, A. (2000). *Changing Minds: Computers, Learning, and Literacy*. Cambridge, MA: MIT Press.

Hayes, E. R., Games, I. E., (2008). Making Computer Games and Design Thinking: A Review of Current Software and Strategies. Games and Culture, 3(3–4), 309–332.

Kay, A. (1991). Computers, Networks and Education. *Scientific American*, 138–48.

Kohen-Vacs, D., Ronen, M., Bar-Ness, O. (2012). Integrating SMS Components into CSCL Scripts. *7th IEEE International Conference on Wireless, Mobile & Ubiquitous Technologies in Education* (WMUTE 2012), Takamatsu, Japan.

Lister, R., Fidge, C., Teague, D. (2009). Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. *SIGCSE Bulletin*. 41, 3 (July 2009), 161–165.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E. (2010). The Scratch programming language and environment. *ACM Transactions on Computing Education*, 10(4), 1–15.

Mareno, N., Bremner, M., Emerson, C. (2010). The use of audience response systems in nursing education: best practice guidelines. *International Journal of Nursing Education Scholarship*, 7, 1–17.

Or-Bach, R. (2013). Higher Education – Educating for Higher Order Skills. *Creative Education*, 4(7A2),17–21.

Papert, S. (1980). *Mindstorms*. New York: Basic Books.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11), 60–67.

Robins, A., Rountree, J., Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13,137–172.

Vos, N., van der Meijden, H., Denessen, E. (2011). Effects of constructing versus playing an educational game on student motivation and deep learning strategy use. *Computers & Education, 56*, 127–137.

Biography

Rachel Or-Bach is a senior lecturer in the Information Management Systems department in Yezreel Valley College. She received her Ph.D. from the Technion. Her main research interest is design of interactive learning environments: representations, interactivity, and intelligent support and adaptation. She publishes in journals of IT for education, Science Education, Computer Science and Information Systems Education, Informing Science etc.