Application of Puzzles to unpuzzle the Programming difficulty through Spoken Tutorial workshops

Kiran L.N. ERANKI[a], **Kannan M. MOUDGALYA**[b]
IDP Educational Technology, IIT Bombay, India
[a]erankikiran@iitb.ac.in, [b]kannan@iitb.ac.in

Abstract: Computer programming is a challenging subject to teach and learn both for students as well as teachers. Cognitive skills such as programming comprehension and debugging are the most important skills necessary for a programmer. Students have a difficulty to build these cognitive skills either due to lack of resources, pedagogy and feedback mechanisms. In order to address some of these challenges spoken tutorial puzzle based approach has been discussed focusing on programming comprehension and debugging skills of the learners. A study has been conducted to analyze the cognitive difficulties of students in programming education through spoken tutorial based workshops. This study comprised of a group of non-CS engineering undergraduates. It is noted that majority of students showed cognitive difficulties related to programming, debugging and program comprehension, irrespective of the programming language used. This paper discusses how puzzles help in building the programming skills of the learners. We also discuss the challenges and benefits of using puzzles to teach programming skills.

Keywords: puzzles, programming, debugging, comprehension, spoken-tutorials, cognition

1. Introduction

Programming involves comprehension and debugging of computer programs using syntax and semantics of different programming languages. Research in programming education shows that the failure rate and drop-out rate of programming courses are relatively high (McCartney., 2004) and overall effectiveness is poor. Learners can be categorized as capable and non-capable based on their programming abilities. Capable students can write programs and they typically learn programming with moderate effort. Whereas, incapable students cannot write correct programs and need more personal attention and cognitive support to learn programming. Since the failure rate is high, the incapable category plays a significant role in the effectiveness of programming course. In this paper, we report the result obtained by assessing comprehension and debugging skills of students using puzzles and visualization tool to teach Java and C/C++ courses through spoken tutorial workshops.

2. Relevance of Comprehension and Debugging Skills

Programming involves several sub tasks and each subtask has critical role to contribute towards programming skill (Neal., 1989). Novice programmers stumble due to cognitive problems at various levels of programming which starts with basic recall or memorization problems, where students fail to remember the keywords, syntax or semantics of a language. And next stage involves comprehension or ability to understand the given programming problem or program code. At this level also students have a serious difficulty to interpret the given code and logically predict the program output.

3. Spoken Tutorial and Visualization Tools

3.1 Relevance of Spoken Tutorials

A spoken tutorial (Moudgalya., 2011) is a screen-cast of an expert demonstrating an activity along with narration. A ten minute spoken tutorial can have more than about one hundred screen transitions. As a result, the screen-cast is the most effective way to create such an instructional material (Eranki., 2012), compared to all other methods, for example, demonstrating how to write a program in Java to store the elements into an array or matrix.

3.2 What is educational in Logical puzzles?

Logic puzzles are becoming more popular among students as they are intriguing and engaging to solve the mystery. Puzzles have also been used for demonstrating the capabilities of logic programming since the beginning of this discipline. We have used logical puzzles designed using flash animation tool to help students visualize the logical flow and programming comprehension of the code. Games and puzzles have a history of engaging aptitude, which will make students gain more repeated exposures required for effective learning. Puzzles on design and analysis of algorithms help students think about algorithms on a more abstract level, divorced from programming and computer language minutiae.

3.3. Visualization Tools

Visualization tools help learners understand logical control flow of the code. They can provide either Intra or inter procedural depending upon the levels of complexity of the program code. (Andrew., 2003). We used two visualization tools namely: *(a).Codeasy*-A java based visualization tool to teach java concepts as shown in Figure 3. *(b).Flash Puzzle*- to teach C/C++ concepts as shown in Figure 2. Codeasy shows 4 sections namely- Methods, constants, expression instance/array sections to visualize the logical flow of the code. Both the tools provide step-wise control of the program code.

4. Research Methodology

4.1. Research Questions: Spoken tutorials along with Animation puzzles have been used to understand the programming competency of the learner. The research questions examined in this study area are: *(a). Does logical puzzles help to build programming comprehension and debugging skills of students? (b).Does programming by visualization improve comprehension and debugging skills?*

4.2. Sample:

The sample comprised 160 non-computer science (CS) students from local engineering college with basic knowledge of computing and Internet skills. A breakup of the sample is given in Table 1. Two randomized groups [A, B] of each 40 students attending either Java or C/C++ subjects were formed. Two hour workshop with pre-post tests and individual assignments for each tutorial were conducted. A total of 9 Java, 9 C/C++ tutorials were used for this study. Experimental Group-A(C/C++) students worked with flash puzzles, while Group-A(Java) students worked with Codeasy tool during the workshop as an assignment, while the control Group-B students worked on the text assignments which was non-interactive and had no- feedback during the workshops.

Table 1: Sample

Group	FOSS	M	F
A	Java C/C++	38 32	2 8
B	Java C/C++	32 30	8 10

FOSS	Group	PreCS	PostCS	Mean-Diff	SD	N
Java	A B	62.33 60.25	172.55 124.55	109.22 64.30	76.84 45.50	40 40
C/C++	A B	64.15 62.35	186.25 141.25	122.10 78.90	86.35 55.50	40 40

Table 2. Competency Analysis

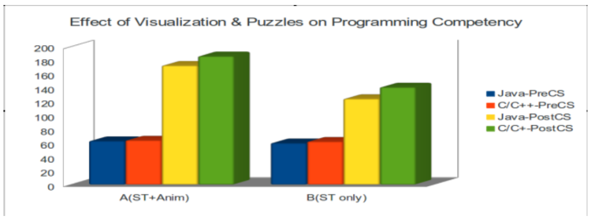
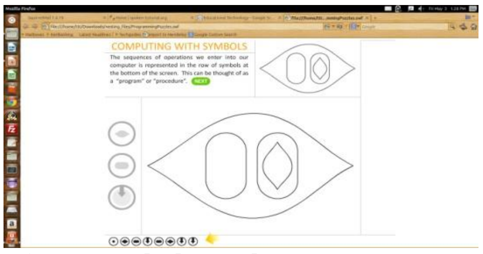


Figure 1: Effect of Visualization on Programming

While Details of competency analysis computed among the groups are shown in Table 2. And similarly, effects of visualization on programming competency is also been shown in Figure 1.

5. Effect of Puzzles and Visualization Tools on Programming competency

We performed quantitative analysis of the pre (PreCS) and post (PostCS) programming competency scores. We computed average gain in comprehension and debugging skills for the whole group as well as sub-groups based on PreCS and gender. We then determined significance using ANOVA and computed the effect size. Feedback scores of 160 participants obtained from pre and post workshops was analyzed. The results of analysis of variance (ANOVA) was significant, F(3,317) = 13.84, p<.0001. The results indicate that PreCS was almost the same for both groups, where as PostCS showed a significant difference among the groups. Group A students of both Java and C/C++ did not differ significantly from one another. This shows that the novice learners improved much more significantly, compared to experienced learners. The improvements were noticed in better visualization, comprehension, comfort to handle programming situations such as writing, debugging and sequencing. As PostCS improved for everyone and for about half the population significantly, we believe that the spoken tutorial based programming along with visualization tool was very effective. Most students report code debugging and understanding of logical flow as a challenge. Programming CDS requires active participation and cognitive thinking skills which needs to be build at the beginning of the course. Our experimental results show that student competency was higher when subjected to visualization tools than non-visual approach used in regular spoken tutorial workshops.



The sequences of operations we enter into our computer is represented in the row of symbols at the bottom of the screen. This can be thought of as a "program" or "procedure".

Figure 2: Flash Puzzle

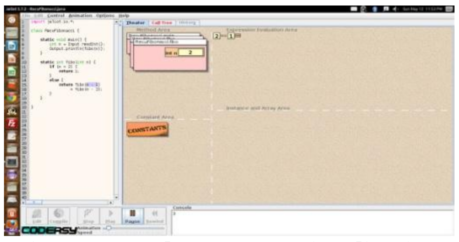


Figure 3: Codeasy- Java visualization tool

6. Conclusions

Using the data collected from students who underwent spoken tutorial workshops on Java, C/C++, effects of visualization tool on comprehension and debugging skills was studied. Novices have found the spoken tutorials to be more useful than those who had prior experience. Visualization tools have helped build mental model and logical flow of the program. Students showed very strong interest to learn programming through visualization and actively participated by themselves. The feedback received from students also talks about the curiosity and interest gained by students, while practicing the programs. While Group B students went through spoken tutorials only without visualization tool
Pedagogical aspects of spoken tutorials and puzzles have also been studied. A statistical approach has been used to discover relationships between programming concepts and visualization tools used to teach students. The study of program comprehension is an interesting discipline to teach and learn. As such, no readily accepted solution exists to validate proposed advances. More standardized benchmark methods are necessary for integrating these tools.

References

Andrew., W. (2003). Observing and measuring cognitive support: Steps towards systematic tool evaluation. *11th IEEE Workshop on Program Comprehension (IWPC'03).* NY, USA: IEEE.

Eranki., K. M. (2012). A Collaborative approach to scaffold Programming efficacy using Spoken Tutorials in Online courses. *8th IEEE conference on CollaborateCom'12.* Pittsburgh, PA, USA: IEEE.

McCartney., R. (2004). A multinational study of reading and tracing skills in novice programmers. *In Working group reports from ITiCSE on innovation and technology in computer science enduser.* ITiCSE, ACM.

Moudgalya., K. M. (2011, Sept 15). Spoken Tutorial: A Collaborative and Scalable Education Technology. *CSI Communications, 35*(6), pp. 10-12. Retrieved from http://www.spoken-tutorial.org/CSI.pdf

Neal., L. R. (1989). A System for example based programming. *Conference on Human factors in computing systems (SIGCHI'89).* (pp. 63-68). ACM.