Wong, L.-H. et al. (Eds.) (2013). Proceedings of the 21st International Conference on Computers in Education. Indonesia: Asia-Pacific Society for Computers in Education

Using Music Notation for Teaching Computer Programming

Eunjeong KO, Kyogu LEE
Music and Audio Research Group, Graduate School of Convergence Science and Technology, Seoul National University, Republic of Korea
{konzung, kglee}@snu.ac.kr

Abstract: Despite the wealth of educational programming languages, many novice programmers face difficulties and give up in the early stages, just because they are not familiar with the programming syntax and semantics. In this paper, we propose a method for programming langauge education using music notation with an aim to entice novice programmers to write their own programs. There are two key aspects in our proposed approach: first, we use music notation as an analogy to programming, based on the observation that there are similar attributes between the two; second, we provide users with on-line auditory feedback to immediately notify potential errors in a pleasant way. We find several key concepts in programming language syntax and semantics, and translate them into music notation to help beginner programmers learn them with ease and intuition. In addition, we design examples and a learning support environment, allowing users to learn to program by themselves.

Keywords: Programming language education, music notation, on-line auditory feedback, learning support environment.

1. Introduction

With respect to education, Wing stated that every educated person in the 21st century will know core computer science concepts, known as computational thinking [1]. The key point of computational thinking is that information and tasks would be processed more systematically and efficiently on the premise that one has knowledge in computer programming. We discuss our study on computer programming education as an expanded concept of computational thinking. However, people who do not write programs regularly face many barriers in the process of learning a programming language. In this study, we focus on using music to facilitate introducing programming concepts for learning programming. A similar attempt was done by Dannenberg who drew analogies between computer programming and music composition in his lecture based on Pascal [2]. The goal was to teach programming to musicians by programmatically creating audio. Meanwhile, most of the methods used for the education of programming languages that are targeted for children and teenagers use visual programming environments and story telling for familiar feedback of programming results [3]. These studies failed to introduce the structure of programming and the debugging processes, which are significant processes of the programming curricular [4].

The goal of our research is to improve the effectiveness of programming education using the method of 'listening' to the program's behavior and utilizing meaningful musical analogies on programming methodology. Music notation is used as an analogy to programming because it contains various similar aspects, such as reusable signs, control statements, and the combinations of sequential and algorithmic structures. In addition, the simplicity of musical symbols and auditory feedback provides an easy and fun learning experience through familiar notations. We investigated users' familiarity in basic western music symbols to assess the potential of using music notation for teaching programming. We created a survey that asked the level of understanding western music scores and experience in music training. A total of 132 participants (76 females, 56 males, mean age = 25.03, SD of age = 2.58) took part in the questionnaire. Most participants stated that they understood western music scores (90.2%), with an average degree of understanding of 3.54 (SD = 1.09), on a scale of 1 to

5. The average number of years of formal music training was 3.75 (SD = 3.28); 5.3% of those with music training had trained for 10 years or more. The results of the survey showed that most people were comfortable with western music symbols and indicated that music could be a good tool for education and interaction.

In this paper, we propose a method to teach basic programming concepts to beginners through the usage of music notation. The novelty of our research is using music and music notation as an analogy to help beginners understand programming concepts and provide on-line feedback on the status of their programs and algorithms. In addition, we propose our approach with a basic programming course involving learning about arithmetic operaion expressions, functions, and its usages related to musical structure. We conduct our research divided in two parts: the concept design of the programming language education and the development of the programming environment.

2. Methodology

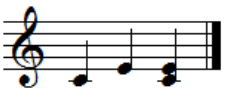
Our research seeks to construct a method for programming education that helps novice programmers gain an understanding in programming logic. As we show in this paper, we provide simple syntax in order to make it easier for beginners to learn the programming concepts instead of trying to interpret complicated code. It is easy to claim that complicated syntax and settings can be overwhelming to novice programmers. Furthermore, a systematic approach should be used when teaching programming concepts, beginning from declaring variables all the way to declaring methods and defining classes [5]. We follow the general curriculum widely used in the computer science field. In this section, we examine the design of programming language education: examples of programming logic, algorithms in a musical setting, and an interpreter for the development environment.

2.1 Examples of Programming Logic

We propose a simplified programming language syntax based on Java. Java is one of the most popular programming languages employed in educational settings, but its complexity makes it difficult for novice programmers. We believe that basic knowledge in programming can be obtained using the methods proposed in our research. Thus, we deal with the following areas of programming logic: variable declaration, arithmetic operation (e.g. addition), and functions (see Table 1).

Table 1: Areas of Programming Logic.

Variable Declaration	Arithmetic Operation	Function

`// note change`
`Sol = Fa + 1;`
`Mi = Re++;`
 	
`// variables as instruments`
`piano, violin, flute;`
 	
`// melody`
`melody() {`
`  Do Re Mi Do;`
`}`
	
	`// notes and addition`
	`Do Mi Do + Mi;`
		`// fibonacci sequence`
		`melody(notes) {`
		`  if(null) {`
		`  null;`
		`  } else if(Do) {`
		`  Do;`
		`  } else {`
		`  melody(notes-1) +`
		`  melody(notes-2);`
		`  }`
		`}`

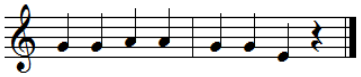
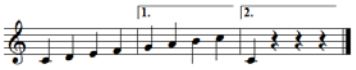
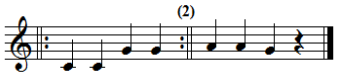
This approach would reduce difficulties and challenges for novice programmers to a certain degree by using musical concepts. Specifically, our development environment provides auditory feedback in on-line of the program and algorithms. For instance, when a novice programmer wants to write a function that creates a Fibonacci sequence, it is difficult to capture the program's behavior. However, on-line auditory feedback enables beginner programmers to recognize the status of the program execution and help them debug the algorithm easier.

2.2 Algorithms in a Musical Setting

For the successful execution of a program, the code needs to be written accurately, based on precise procedures and methods. We establish a set of basic algorithms with musical analogies for tutoring computer programming: sequential execution, conditional structure, and repetition (loops) (see Table 2). The relationships between the two help beginners understand programming concepts easily.

Table 2: Basic Algorithms with Musical Analogies.

Sequential Execution	Conditional Structure	Repetition (Loops)


`melody() {`
`  Sol Sol La La;`
`  Sol Sol Mi Rest;`
`}`
	
	`melody() {`
	`  Do Re Mi Fa;`
	`}`
	`if(melody once) {`
	`  first ending;`
	`} else {`
	`  second ending;`
	`}`
		
		`repetition(first bar, twice){`
		`  melody(first bar) {`
		`  Do Do Sol Sol;`
		`  }`
		`}`
		`melody(second bar) {`
		`  La La Sol Rest;`
		`}`

2.3 Interpreter for Development Environment

Our prototype provides an interpreter (or Java application) as the development environment using JFugue. JFugue is a Java API used to play music with code and is used to 'play' a program in our approach. With simple syntax, novice programmers write their program and listen to the results immediately through the interpreter. This makes it possible to notify the programmer of errors and aids users in fixing them. There is more work that needs to be done to design the interpreter with syntax highlighting and dialogs for finding examples dynamically.

3. Conclusion

We believe the effects of using music notation as an analogy for programming education will improve the performance and comprehension of novice programmers. We are planning to evaluate our methodology and the following hypothesis will be investigated: Our research can assist novice programmers in enabling to write programs and identify the program's flow easily in the learning process. Participants who do not have any programming experience would be selected and split into two experimental groups. In the control group, they might take lessons how to write a program with a high-level programming language. In contrast, in the experimental group, they might study how to write a program based on our approach. We will analyze the quantitative data of the experiments based on log files and survey responses collected from the participants. The survey may ask about the process of finding bugs, notifications of the program's behavior, level of achievement of the goal, and the quality of the results and program usability. We propose the idea of teaching computer programming, which could explain the programming process more easily and efficiently. In the next step, we will focus on the evaluation and the implementation of our method. Our research would bring a more pleasant and enjoyable environment for programming language education.

References

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.

Dannenberg, F. K., Dannenberg, R. B., & Miller, P. L. (1984). Teaching Programming to Musicians.

Utting, I., Cooper, S., Kölling, M., Maloney, J., & Resnick, M. (2010). Alice, greenfoot, and scratch—a discussion. ACM Transactions on Computing Education (TOCE), 10(4), 17.

Lee, M. J., & Ko, A. J. (2011). Personifying programming tool feedback improves novice programmers' learning. In Proceedings of the seventh international workshop on Computing education research (pp. 109-116). ACM.

Hughes, D. S. (2012). Introducing programming logic in a one-credit course. In Proceedings of the 50th Annual Southeast Regional Conference (pp.48-52). ACM.