Suggesting and supporting examples for novice programmers

Michelle Ichinco
Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO
michelle.ichinco@wustl.edu

I. Introduction

While computer science education has recently begun expanding in middle and high schools, many students still do not have access to computer science education in a classroom [1]. As a result, many children learn programming outside of formal education using novice programming environments and games such as Scratch [2] and Code.org [3].

For these independent learners, there are various ways to gain skills, such as through tutorials, games, or remixing. However, a novice programmer working on their own project currently has no way to receive feedback on their program, like they might in a classroom. Furthermore, since a novice programmer likely does not know the scope of possibilities, they will not necessarily know what questions to ask to advance their knowledge of Application Programming Interfaces (APIs), languages, or programming skills.

One way to help novice programmers improve their programming skills while they work on open-ended projects might be to suggest context-relevant feedback as novices program. This work presents the Example Guru, a way of suggesting supported examples to novices with the goal of increasing their exposure to programming skills. The key components of the Example Guru are 1) the idea of suggesting context-relevant code examples during open-ended programming 2) support for helping novices to understand and learn from code through explanatory examples.

To give an example of how this would actually work in practice, imagine a novice programmer, Annie. She is creating a 3D animation with a flamingo doing a simple turn in Looking Glass [4], a blocks-based programming environment. The Example Guru discovers the simple turn and triggers a suggestion for a more complex dance animation, as shown in Fig. 1. When Annie notices the suggestion about her flamingo doing a more complex dance animation, she wants to know more and decides to explore the suggestion. The suggestion has example code and a video showing how it works. Annie watches the video, but does not know how to make the complex dancing animation, so she looks at the code. It has concepts that she has not seen before, but the Example Guru provides support that helps her understand how the code works. Enabling the Example Guru to do this requires a better understanding of what support actually helps novices

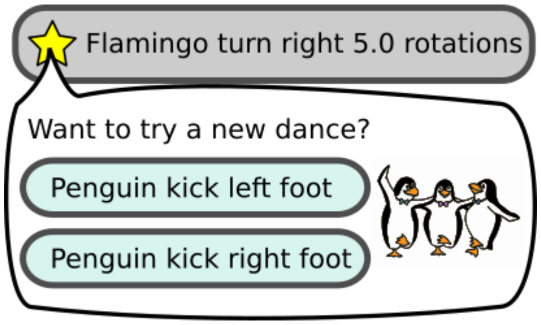

Fig. 1. The user receives a suggestion for a more complex dance animation.

to understand example code for concepts they are unfamiliar with. Finally, in an ideal scenario, Annie uses the concept from the example and then continues to explore the suggested programming constructs as she programs.

I will first describe the prior and preliminary work towards this goal, followed by the future work necessary to complete it.

II. Prior and and preliminary work

I have completed two exploratory studies that provide critical support for the Example Guru: one about how to suggest feedback and another on understanding the way novice programmers currently use examples. I have also completed preliminary work supporting the idea of suggesting simple examples as a way of introducing novice programmers to unfamiliar API features while programming.

A. *Suggesting code improvements*

In a prior study, I explored the types of suggestions experienced programmers made to novice programs [5]. The goal of this study was to understand the styles of feedback that a novice programmer might receive in a natural mentoring relationship with a friend or family member who is a more experienced programmer. This feedback was often focused on improving the coding quality, but sometimes also focused on the output of the code. For our feedback, we decided to integrate these two styles, ensuring that the suggested feedback introduces a new concept or a better way to use a skill, but also fits into the context of the animation. I believe that having a suggestion relate to the animation (or other output) is critical for motivation in the context of child novice programmers.

B. Exploring examples

Examples have been used effectively in education [6] and are also a popular way for all levels of programmers to fix bugs or learn new things [7]. Studies have hinted that novice programmers have trouble using examples [8], but had not explored what was causing difficulties. I ran a study looking at novice programmer example use. The study found a number of obstacles to successful example use, from example comprehension, to issues locating blocks in the programming environment [9]. Furthermore, novices were successful when they continued referencing an example throughout a programming task and when they continued generating new ideas to try to solve the task.

C. The Example Guru: demonstrative examples

I am currently working on the design and evaluation of the Example Guru for API methods. Testing suggested feedback with API methods allows us to focus on the suggesting aspect, as only minimal examples are necessary. For API methods, we can provide demonstrative examples, which are simple 1-3 line examples with the purpose of showing what each API method does. This is in contrast to the explanatory examples I will discuss later, which help novices learn complex new programming skills.

There are three main components of the system, which have been designed based on iterative formative one-on-one testing: 1) the rules that determine when a user should receive a suggestion, 2) the suggestion content and design, and 3) the examples. Rules check novice programs to determine whether a novice program has an opportunity to use a certain API method. The rules are based on differences we found in novice and expert programs. When a novice programmer executes their program, the Example Guru checks for 'passing' rules, meaning there is an opportunity for a suggestion. The user receives only one suggestion at a time to reduce the risk of overwhelming them. Each suggestion has a text description and a preview of the example code execution. If a user decides to open a suggestion, they can view two contrasting examples with code and the ability to execute the code.

I ran a small pilot study in which users were introduced to suggestions and allowed to work on projects on their own for 30-40 minutes. Novice programmers were receptive to suggestions, exploring them and often using the suggested API methods within their programs without prompting. This part of the system still requires a formal evaluation of the impacts of suggested examples on novices' programming behavior.

III. Next Steps

My preliminary work shows the potential for suggestions as a way to support novice learning, but also reveals challenges in using complex code examples. When examples are more complex, novices will be more likely to struggle in actually using and learning the programming concepts [8]. Complex in this sense means that the concept requires multiple lines of code and that the example must actually explain the concept to the programmer. I hypothesize that studying the differences in the ways experts and novices use examples will provide the necessary information to design support for complex examples.

A. Understanding novice and expert example use

I propose running a study comparing novice and expert programmers' example use for complex examples. I believe the results will provide critical insight into how to better assist novice programmers in using examples. Related work in chess discusses how experts unknowingly "chunk" information that they are processing, meaning that they can process a number of chess pieces as one element due to their expertise [10]. Similarly, I hypothesize that expert programmers will comprehend complex examples differently than novices in a way that a system might be able to mimic to support complex example use.

B. The Example Guru: explanatory examples

Finally, based on the hurdles and strategies from prior work and the comparison of novice and expert example use, I will design explanatory examples for complex programming concepts. The example support will be integrated into the Example Guru in order to suggest advanced programming concepts to novice programmers working on open-ended projects. If successful, the Example Guru will suggest useful and relevant examples of varying complexity, which will increase novice programmers' knowledge of APIs and programming concepts.

References

[1] Google, "Searching for Computer Science: Access and Barriers in U.S. K-12 Education," 2015. [Online]. Available: http://services.google.com/fh/files/misc/searching-for-computer-science_report.pdf

[2] "Scratch - Imagine, Program, Share." [Online]. Available: https://scratch.mit.edu/

[3] "Anybody can learn | Code.org." [Online]. Available: http://code.org/

[4] "Looking Glass Community." [Online]. Available: https://lookingglass.wustl.edu/

[5] M. Ichinco, A. Zemach, and C. Kelleher, "Towards generalizing expert programmers' suggestions for novice programmers," in *Visual Languages and Human-Centric Computing (VL/HCC), 2013 IEEE Symposium on*. IEEE, pp. 143–150.

[6] J. Sweller and G. A. Cooper, "The use of worked examples as a substitute for problem solving in learning algebra," *Cognition and Instruction*, vol. 2, no. 1, pp. 59–89, 1985.

[7] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer, "Two studies of opportunistic programming: interleaving web foraging, learning, and writing code," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009, pp. 1589–1598.

[8] M. B. Rosson, J. Ballin, and H. Nash, "Everyday programming: Challenges and opportunities for informal web development," in *Visual Languages and Human Centric Computing, 2004 IEEE Symposium on*, 2004, pp. 123–130.

[9] M. Ichinco and C. Kelleher, "Exploring novice programmer example use," in *Visual Languages and Human-Centric Computing (VL/HCC), 2015 IEEE Symposium on*. IEEE, 2015, pp. 63–71.

[10] A. D. De Groot and A. D. de Groot, *Thought and choice in chess*. Walter de Gruyter, 1978, vol. 4.