Suggesting Examples to Novice Programmers in an Open-Ended Context with the Example Guru

Michelle Ichinco, Wint Hnin, and Caitlin Kelleher
Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO, USA
{michelle.ichinco, hnin, ckelleher}@wustl.edu

***Abstract*—Many novice programmers use blocks-based programming environments outside of classrooms, due to a lack of computer science education in schools. Many solutions for supporting learning in these environments are out of the context of the programming environment and the user's project. In this poster, we present a way of expanding novice users' knowledge of a blocks-based programming environment by suggesting examples during open-ended programming.**

I. Introduction

Current systems for novice programmers often provide some support for learning, such as games, tutorials, or remixing. In these systems, novices are either constrained to specific task goals, or must seek out opportunities for learning when programming without constraints. One such area where novices will likely have trouble is API usage, where they may not be aware of relevant API methods that would improve their programs. Interestingly, this problem also applies to more experienced programmers learning APIs, as they are also often unaware of all possible methods.

In this poster, we introduce the Example Guru, a system that provides ideas of ways a programmer could use an unfamiliar or incorrectly used API method when programmers are working in an open-ended context without a system-defined task. The Example Guru provides examples to demonstrate how each suggested method can be used.

II. Related Work

Related work includes: suggested examples for novice programmers and software engineers and help for new users of complex systems. When novice or end-user programmers work on specific tasks, current systems suggest example code to fix errors [1], [2]. In an open-ended context, experienced programmers benefit from suggested examples for API usage [3], [4]. Suggested information also assists novices using complex software, like photo editing software [5], [6]. However, the Example Guru focuses on suggesting relevant code examples to novices on open-ended projects to improve API knowledge, rather than fixing errors.

III. The Example Guru

In this work, we will discuss a version of the Example Guru that aims to improve a user's knowledge of the Looking Glass API by suggesting unfamiliar API methods or familiar methods in new ways. Looking Glass is a blocks-based programming environment for middle school children, in which users create 3D animations.

The Example Guru contains a set of 'rules', which are programs that determine what to suggest to the programmer and when to suggest it. The 'suggestions' appear in the programming environment as text descriptions (see Fig. 1-A). Upon selecting a suggestion to explore, the novice programmer can view two contrasting 'examples', which contain code, the ability to execute the code, and information about where to access the API call in the programming environment (see Fig. 1-C,E,D,F). The next three sections describe in more detail the design of the rules, suggestions, and examples.

A. Rules

Rules are comprised of internal code that tests a novice program for a certain pattern. If the rule finds the pattern in the novice program, it triggers a suggestion. In this case, the goal of the Example Guru is to expose novice programmers to API methods that they likely did not realize existed or did not know how to use correctly. Thus, the set of rules finds situations where a novice program would benefit from information about a certain API method. For this work, the authors created rules manually, but we believe that the rule creation process could be partially automated or crowd-sourced as well.

B. Suggestions

In designing the suggestions, we had two main goals: 1) do not interrupt the user, and 2) make it easy for users to decide whether or not to use suggestions. To not interrupt users, the Example Guru only adds one new suggestion and only adds suggestions when the novice programmer executes their code. We chose this time to present suggestions because it likely means that users have finished an idea in the code and want to see how it works. The suggestions are added to the interface in a list (see Fig. 1-A), as well as to the code blocks themselves (see Fig. 1-B). Users can then choose whether they want to explore the suggestions, as they are not required to interact with them in any way. We also wanted users to be able to easily decide whether or not to use a suggestion. In order to accomplish this, the suggestions have a text description and

978-1-5090-0252-8/16/$31.00 © 2016 IEEE

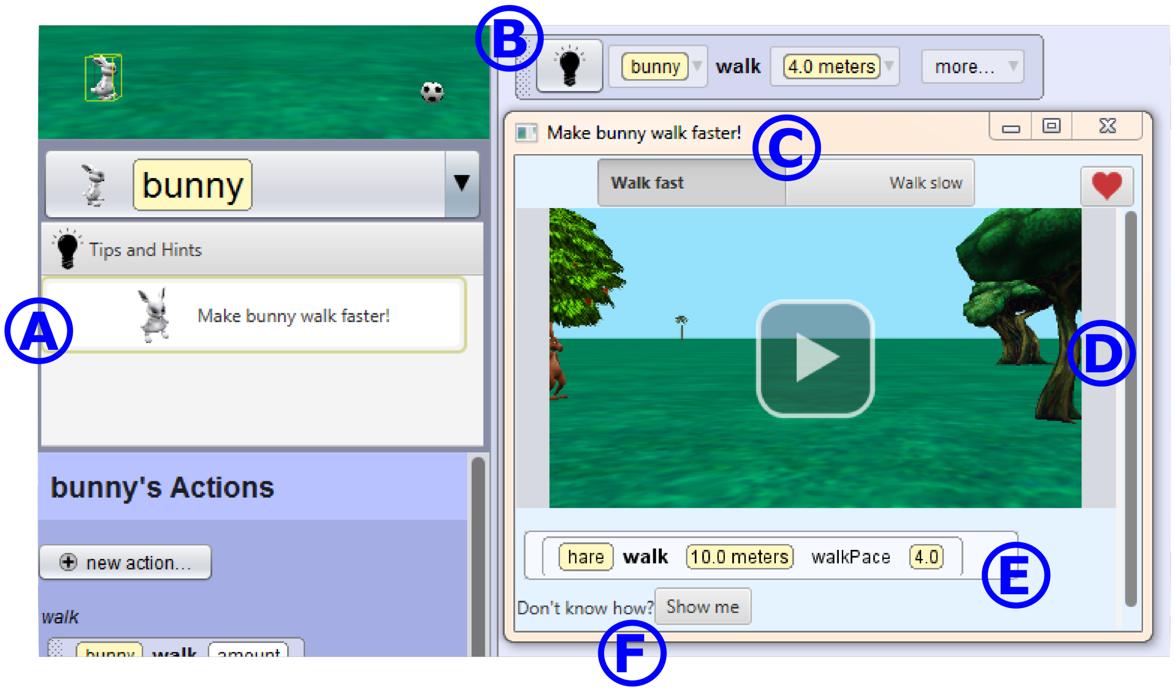

Fig. 1. A cropped screenshot of the Looking Glass interface with a suggested example open. (A) Suggestions appear in a list. When suggestions are clicked on, the examples open. (B) Users can open a suggestion from a button on a code block. (C) The suggestion has two contrasting examples. (D) Users can watch the example execute. (E) Example code is shown. (F) A user can click the 'show me' button to get help finding the API method in the example.

also show a preview of the example if a user hovers over the suggestion. In early formative testing, participants opened the suggestions from both the list of suggestions and the code and felt comfortable with the pace of the new suggestions added.

C. *Examples*

We wanted examples to: 1) clearly demonstrate the API method, 2) use contrast to emphasize the API method, and 3) enable the user to easily find the API method code block. We created each example with one to three lines of code to focus on the API element being suggested and decrease extra complexity that might make the example confusing. The two examples are also designed to show contrast in order to clarify the meaning of the API method being introduced. For example, for the suggestion introducing 'walkPace', the two examples show a fast and slow 'walkPace' (see Fig. 1-C). Finally, we also added a 'show me' button to help users locate the suggested API method in the interface (see Fig. 1-F).

IV. Future Work

Our next step is to compare the Example Guru to ways novices currently learn new API methods. We also intend to expand this system to support more complex concepts.

A. *Comparison and long term*

Two main questions we have about this system are: 1) whether novices will continue to use the Example Guru beyond their first session and 2) how the Example Guru compares to other ways novices learn programming. We would like to understand more long-term usage of the Example Guru in order to understand whether novices continue to benefit from it, as well as to rule out the novelty effect. We will also compare the Example Guru to other current ways that novice programmers learn, such as documentation.

B. *Example Types*

While we decided to first populate the Example Guru with suggestions for API methods, we believe the system could also be used for other purposes. The Example Guru might be able to introduce more complex programming concepts, like parallel execution and loops. Another possibility would be for the Example Guru to aid in motivating learners by introducing fun animations or code.

References

[1] B. Hartmann, D. MacDougall, J. Brandt, and S. R. Klemmer, "What would other programmers do: suggesting solutions to error messages," in *Proc. conf. on Human factors in comput. syst.*, 2010, pp. 1019–1028.
[2] W. Jernigan, A. Horvath, M. Lee, M. Burnett, T. Cuilty, S. Kuttal, A. Peters, I. Kwan, F. Bahmani, and A. Ko, "A principled evaluation for a principled idea garden," in *Visual Lang. and Human-Centric Comp. , 2015 IEEE Symp. on.* IEEE, 2015, pp. 235–243.
[3] H. Zhong, T. Xie, L. Zhang, J. Pei, and H. Mei, "MAPO: Mining and recommending API usage patterns," in *ECOOP 2009Object-Oriented Programming*. Springer, 2009, pp. 318–343.
[4] R. Holmes, R. J. Walker, and G. C. Murphy, "Approximate structural context matching: An approach to recommend relevant examples," *Software Eng., IEEE Tran. on*, vol. 32, no. 12, pp. 952–970, 2006.
[5] B. Lafreniere, P. K. Chilana, A. Fourney, and M. A. Terry, "These aren't the commands you're looking for: Addressing false feedforward in feature-rich software," in *Proc. of the ACM Symp. on User Interface Soft. & Tech.* ACM, 2015, pp. 619–628.
[6] J. Matejka, T. Grossman, and G. Fitzmaurice, "Ambient help," in *Proc. of theConfer. on Human Factors in Comp. Sys.* ACM, 2011, pp. 2751–2760.