The Impact of Distractors in Programming Completion Puzzles on Novice Programmers Position Statement

Kyle J. Harms
Department of Computer Science & Engineering
Washington University in St. Louis
St. Louis, Missouri, United States
kyle.harms@wustl.edu

I. Position Statement

Our previous work has demonstrated that programming completion puzzles enable novice programmers to acquire new programming skills [1]. As shown in Fig. 1, programming completion puzzles ask users to reassemble a block-based program's statements into the correct order. Users use the available blocks in the puzzle statement bin (Fig. 1-A) and place them into the correct order in the puzzle workspace (Fig. 1-B). In our previous work we only included blocks that were part of the actual puzzle's solution [1]. However other puzzle-like programming systems often include distractor statements as part of the user's experience [2]. In the context of programming puzzles, distractors are extra blocks or statements that are not part of a puzzle's solution. We wondered what impact distractor statements might have in programming completion puzzles on novice programmers? Do distractors also help facilitate learning programming skills when used in programming completion puzzles?

Our programming completion puzzles are heavily inspired by Cognitive Load Theory's completion problems [3]. Completion problems are partially completed worked examples where a user completes the remainder of the example. By limiting extraneous work, completion problems focus a learner's mental resources towards processing new material. Our completion puzzles limit the blocks users need to use to complete the problem. This is intended to focus and direct their mental resources towards learning new programming concepts; a learner does not need to expend mental resources navigating the interface to the find the correct block. However, we were unsure how including distractors into completion puzzles might affect learning outcomes. Since the intent of these problems is to carefully manage learning resources, it is possible that distractors may overwhelm learners' limited working memory resources, thereby limiting their ability to learn new programming concepts.

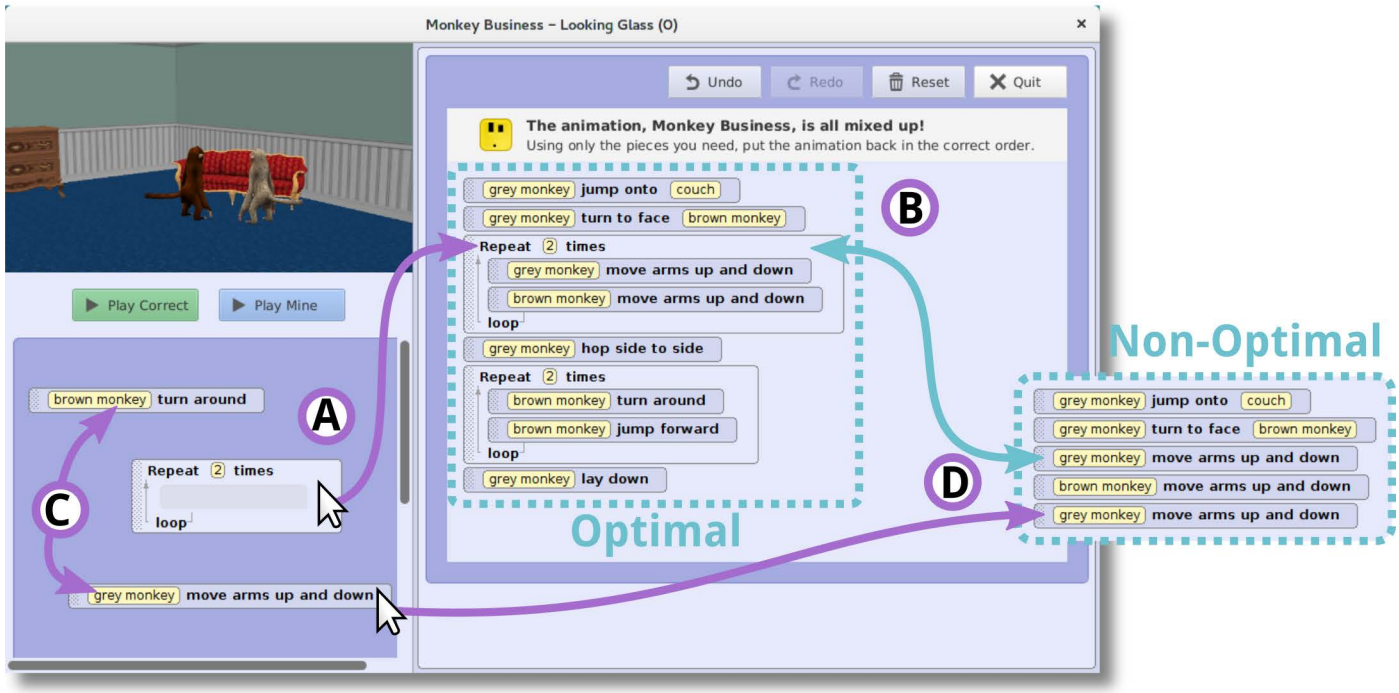

Fig. 1. A typical programming completion puzzle with distractors. Users complete these puzzles by dragging blocks from the puzzle statement bin (A) and dropping the blocks into the puzzle workspace (B). This puzzle includes several distractor statements (C) and an example of their use in a non-optimal solution (D).

978-1-4673-8367-7/15/$31.00 ©2015 IEEE

Recently, we have been conducting a formative evaluation into the impact of distractors within programming completion puzzles. Fig. 1-C shows an example of several distractor statements. Distractor statements are not part of the solution to the puzzle, instead they are extra unnecessary statements that user's must rule out in order to solve a puzzle.

From our early observations in this evaluation, we have noticed that distractors seem to make the puzzles more challenging for users. We have observed a greater frequency of comments related to the distractor puzzles being more challenging and more fun. Further, users typically need more time to complete a puzzle with distractors than one without. If distractors do increase the difficulty of the puzzle and the motivation to remain engaged with the puzzles, then they might provide a way to encourage users to continue practicing non-mastered programming concepts.

So far in our formative evaluation we have noticed two ways to use distractors to increase the difficulty of the puzzles: 1) use distractors to encourage users to follow a sub-optimal path, and 2) mix up the type of distractors between puzzles. Both of these methods appear to increase the challenge of the puzzle while also encouraging users to pay closer attention to the material.

We use sub-optimal path distractors to encourage users to begin to construct an alternative and incorrect solution to a puzzle. When constructing an alternative solution, the user will fail to fully complete the alternative solution because the puzzle is missing several critical pieces necessary for that solution. Once a user realizes that she cannot complete the puzzle using the alternative solution, she is forced to reevaluate her solution strategy to come up with the proper and correct solution.

See Fig. 1-D for an example of a distractor which encourages users to take a sub-optimal path. In this puzzle, we noticed that novices frequently want to repeat statements by duplicating the statements instead of using a loop. When we provide the distractor, "grey monkey move arms up and down," they usually start down the sub-optimal *duplicate statements* path. However, they eventually realize they cannot complete this puzzle by duplicating the statements because they lack the final statement necessary for this solution: "brown monkey move arms up and down." Rather, they have to reevaluate their solution to realize that they must use a loop instead. It appears that leading users down a sub-optimal path has the effect of making the puzzle more challenging while possibly encouraging more practice with programming concepts.

We have also observed that providing an unpredictable experience by mixing up the type of distractors between puzzles keeps users alert. For example, a puzzle curriculum might begin with a puzzle that includes sub-optimal distractors and then follow up with a puzzle that has no distractors. While completing the second puzzle, we have observed that participants carefully consider whether or not each statement is necessary for a solution. With a predictable experience, the users may begin to use their expectations about distractors to simplify the problem solving process. An unpredictable completion problem experience may encourage users to pay closer attention to the elements needed to solve each puzzle.

Our early work suggests that distractors may provide an additional approach to encourage novices to learn and practice programming skills. We also think that the additional challenge introduced by the distractors may increase the longevity of programming puzzles as a tool novices may use to develop their programming skills.

REFERENCES

[1] K. J. Harms, N. Rowlett, and C. Kelleher, "Enabling Independent Learning of Programming Concepts through Programming Completion Puzzles," in *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2015.

[2] M. J. Lee, A. J. Ko, and I. Kwan, "In-game Assessments Increase Novice Programmers' Engagement and Level Completion Speed," in *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, New York, NY, USA, 2013, pp. 153–160.

[3] J. J. G. Van Merriënboer, "Strategies for Programming Instruction in High School: Program Completion vs. Program Generation," *J. Educ. Comput. Res.*, vol. 6, no. 3, pp. 265–285, Jan. 1990.