Learning Loops: Affordances and Challenges of Project Bloks

Veronica Lin, Stanford Graduate School of Education, vronlin@stanford.edu
Paulo Blikstein, Stanford Graduate School of Education, paulob@stanford.edu

Abstract: Programming for young children has gained significant traction in recent years. Although many learning tools have been created toward this end, research has focused primarily on older children, and few studies have documented young children's learning processes and difficulties during programming activities. Through task-based studies with first-grade children, we examine the design affordances of a tangible user interface, Project Bloks. For the scope of this paper, we focus on two coding puzzles where children engage with loops. Our findings reveal that the design affordances of the system are conducive to the learning of loops, and highlight some difficulties that children encounter with loops. We discuss the implications of these preliminary results for designers and educators, and propose ideas for future work.

Introduction

In recent years, efforts to teach computer programming in formal and informal learning environments have steadily grown. Programming for K-12 was first introduced in the 1960s with Logo (Papert, 1980), and we have seen a new wave of interest in promoting computational literacy (diSessa, 2000) and introducing computational thinking (CT) to K-12 students (Wing, 2006; Blikstein, 2018). This has been accompanied by the development of programming environments specifically for children, including visual programming languages such as Scratch (Resnick et al., 2009) as well as tangible user interfaces for very young learners, such as KIBO (Bers et al., 2013).

Despite the abundance of child-friendly tools for learning programming, there is little research on the use of such tools with young children and even less on the thinking and learning processes during programming activities (Lye & Koh, 2014; Fessakis et al., 2012; Bers et al., 2013). We conducted task-based studies with children using Project Bloks, a novel tangible programming platform designed by members of our team in partnership with industry, and digital coding puzzles. We focus on the last 2 puzzles, where children engage with loops, a type of control flow instruction and a key concept of CT (Brennan & Resnick, 2012). Our study investigates whether the affordances of the system are conducive to the perception of need and learning loops, and the types of difficulties children may encounter when learning about loops.

Methods

We conducted individual, task-based studies with 12 first-grade students (6 girls, 6 boys) at a suburban elementary school. We focus on the last 2 puzzles only (P10 and P11), where complex loops were required. Children used Project Bloks, a set of physical blocks and icon-based symbols, to program a "bee" avatar to navigate through a digital maze. The icons represent instructions for directions (top, down, left, right) and commands (get flower, make honey). The repeat functionality is expressed with open/close blocks that resemble opening and closing brackets, and physically encompass the blocks to be repeated. After a brief discussion about building activities, children were solved the series of 11 puzzles with limited guidance from the researcher. Puzzles were presented in order of increasing difficulty; at P8, the "repeat" blocks were introduced. After the puzzles, children were asked to reflect on their experience using a smiley-o-meter and a few additional questions.

Data and findings

We video-recorded and transcribed all sessions and captured all intermediary versions of the code at runtime. All 11 children (5 girls, 6 boys) who attempted P10 completed it, with an average of 4 runs (range: 1, 9) and an average duration of 4m28s (range: 1m43s, 8m54s). For P11, all 7 children (4 girls, 3 boys) who attempted it were successful, with an average of 4 runs (range: 1, 8) and an average duration of 3m34s (range: 1m55s, 6m05s).

Studies revealed that the limited number of physical blocks creates an opportunity for children to engage with loops, in a way that on-screen programming languages do not. We observed that indeed the limited-block design prompted children to think harder upon realizing that the full sequence could not be physically expressed with the available blocks. 6 of 11 children in P10 and 2 of 7 children in P11 made explicit verbalized a need for more blocks; i.e., "*I don't have enough blocks*" or "*Wait, we need more pads*!" When faced with this limitation, children started to consider solutions with control flow structures rather than just single-action blocks.

A common difficulty we observed resulted from the simple introduction of the "repeat" block. Because the prior puzzles initially repeated only a single command, 5 of 11 children were unsure about whether they could use the repeat with more than one command in P10, which needed to repeat a sequence of 4 commands.

An exemplary case of problem-solving is Mallory, who begins by placing blocks for the full sequence (Fig. 2a). Upon noticing that there are not enough blocks to continue this way, she asks, "*Does the repeat block do like…two different ones [blocks]? Or just one?*" Her inquiry demonstrates how the limited-block design triggered the need for a different type of control flow structure. She initially places the open and close repeat blocks around the first two blocks (Fig. 2b) and says, "*I don't think that would work. Because then I would keep repeating these, and not do any of these.*" Without further prompting, she puts the sequence of 4 blocks together (Fig. 2c) and places one hand each on the open and close repeat blocks. The simultaneous action of both hands demonstrates how the bracket design maps to Mallory's gestures. She is able to then run the program successfully, suggesting that the design affordances of Project Bloks are conducive to the learning of loops; the limited-block design requires her to use a loop where she otherwise might not have thought to at that point.

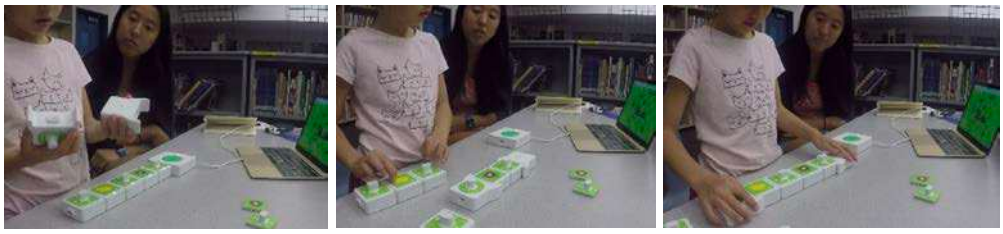

Figure 2a, 2b, 2c. Mallory solving P10.

Discussion

These studies, and our preliminary analysis of P10 and P11, illustrate the potential for children to learn complex CT concepts, such as loops, using tangible programming interfaces like Project Bloks. The limited-block design is a contribution to the research on programming tools for children; while it is intrinsic to tangible user interfaces, it is missing from digital programming languages, where blocks are of infinite quantity. The system demonstrates how the practical limitation of blocks can be turned into a feature, productively triggering the need for a loop structure and creating opportunities for children to learn a key CT concept.

We also described a key difficulty that children encountered with loops. Although we often assume that children should progress from simple to complex, we found evidence that this may not be the case in programming activities. Because the repeat block was introduced in a context that repeated only a single command, there was no indication for children that multiple commands could be repeated. Thus, when introducing a CT concept such as "loops," it is important to consider how the first action may bias users and prevent them from thinking of other ways to use it. We hypothesize that introducing loops with multiple blocks may be more productive, and that this principle of "complex first" might also apply to other constructs, such as conditionals and functions.

We acknowledge that our results are promising but also preliminary. We focus here on loops; however, our findings and larger dataset reveal patterns that may be applicable to other CT concepts, such as functions.

References

Bers, M., Flannery, L., Kazakoff, E., Sullivan, A. (2013). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education*, 72, 145-157.

Blikstein, P. (2018). Pre-College Computer Science Education: A Survey of the Field. Mountain View, CA: Google LLC. Retrieved from https://goo.gl/gmS1Vm

Brennan, K., Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Annual Meeting of AERA 2012*.

diSessa, A. A. (2000). Changing minds: Computers, learning, and literacy. Cambridge: MIT Press.

Fessakis, G., Gouli, E., Mavroudi, E. (2012). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. *Computers and Education*, 63, 87-97.

Lye, S., Koh, J. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51-61.

Papert, S. (1980). Mindstorms. Children, computers and powerful ideas. New York: Basic Books.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11), 60-67.

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

Acknowledgements

This work was supported by a Google Faculty Award.