Learning Loops with Tangible Programming Interfaces for Children

Final Report for the Amir Lopatin Fellowship, 2017 - 2018

Veronica Lin, 2nd year PhD in Learning Sciences and Technology Design / Developmental and Psychological Sciences

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, I examine how a tangible user interface called Project Bloks helps children develop computational thinking skills. For the analysis, I focused specifically on two coding puzzles where children engage with the fundamental computational thinking concept of loops, with the following research questions:

(1) Are the affordances of Project Bloks conducive to the perception of need and learning loops?
(2) What kind of difficulties do children encounter when learning about loops?

The findings reveal several design affordances and challenges of using educational technology to offer meaningful learning experiences for young children. Given that tangible programming interfaces are a novel and compelling platform for helping children learn, this work has important implications for designers and educators.

INTRODUCTION

"*Sometimes I thought that it was very hard to put things together, but it's actually quite easy*." This was a ten-year-old girl's reflection following a two hour "Girls in Action Workshop" that I led in Cape Town, South Africa in July 2015 to introduce young girls to robotics. There, and in my other research and teaching experiences, I realized the potential of tangible and digital technologies like Scratch, littleBits, and KIBO to help children learn within and beyond classroom walls.

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), a programming language designed primarily to teach mathematical concepts. Logo was widely adopted in schools during the 80s and 90s, but was not long-lasting. Since the potent idea of computational literacy arose (diSessa, 2000), we have seen a renewed interest in introducing programming and computational thinking (CT) to K-12 students (Wing, 2006). 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 (Sullivan et al., 2017). Industry has joined the efforts to appeal to young consumers (ages 4+) with a number of iPad apps (e.g. Hopscotch,

Kodable, Scratch Jr.), building sets (e.g. littleBits, LightUp, Roominate), and programmable robots (e.g. KIBO, Dash & Dot, Sphero).

For young children, tangible programming interfaces are particularly compelling as a learning platform. Because the interaction style of tangible toys is already familiar to young children, they offer a low threshold for participation and are believed to be both accessible and beneficial for learners ages 4 to 7 (Minogue, 2006). Researchers have suggested that tangible user interfaces (TUIs) – user interface technologies that link the physical and digital worlds – can enhance learning because they promote active, hands-on engagement; offer opportunities for collaboration between users; and allow for exploration and reflection (Xu, 2005; O'Malley, 2004). As contexts for learning increasingly extend outside of schools and into homes and community centers, especially with the emergence of affordable technologies, TUIs have become more pervasive as learning tools for young children.

Despite the abundance of child-friendly tools for teaching and 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. A review on empirical research of CT in K-12 contexts (Lye & Koh, 2014) demonstrates a heavy focus on children ages 9 and up; of the 27 intervention studies analyzed, only 2 were with young children. Fessakis et al. (2012) presented an exploratory case study of a kindergarten class using a Logo-like programming interface and an interactive whiteboard. They described children's difficulties with usability and orientation, and observed two problem-solving strategies: planning and trial-and-error. Bers et al. (2013) used quantitative methods to evaluate learning outcomes on CT concepts in three kindergarten classrooms. They found evidence that their curriculum fostered engagement and learning, and noted lower achievement with control flow instructions, such as loops and conditionals. This work builds on prior research, examining the design affordances and challenges of a tangible programming interface. Although I focus only on the concept of loops, similar methods can be used to address other computational thinking concepts.

From an educational perspective, it is important to understand not only what children are doing during programming activities, but also why and how. I conducted task-based studies with children using Project Bloks, a novel tangible programming platform designed by Paulo Blikstein in partnership with industry, and a series of 11 maze-based coding puzzles. Here, I 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). Although loops are supported in many digital blocks programming languages, such as Blockly and Scratch, as well as tangible programming interfaces, such as KIBO (Sullivan et al., 2017), Dr. Wagon (Chawla et al., 2013), and Tern (Horn & Jacob, 2007), little research exists to study the learning affordances and challenges associated with the design of the "repeat" blocks, the simplest form of loops. Loops are significant because they are intrinsically computational; they are the first type of more elaborate computational structures that children encounter after they have mastered the simpler, more intuitive single-action commands, such as "forward" or "turn right." This study investigates the following research questions:

(1) Are the affordances of Project Bloks conducive to the perception of need and learning loops?
(2) What kind of difficulties do children encounter when learning about loops?

METHODS

Participants

I conducted individual, task-based studies with 12 first-grade students (6 females, 6 males) at a suburban elementary school. In this paper, I report on the last 2 puzzles only, where more loop-related activities arose; students who did not attempt those puzzles due to time constraints are excluded from these analyses. 11 children (5 females, 6 males) attempted Puzzle 10, and 7 children (4 females, 3 males) attempted Puzzle 11.

Materials

Project Bloks is a tangible programming platform that creates coding experiences to help young children learn programming. Many systems are compatible with Bloks, including Lego WeDo, MiroBot, and screen-based programming mazes based on code.org activities. The modularity, tangibility, and open-source nature of Project Bloks makes it a unique computational toolkit. In this study, I used the latter, in which children use physical blocks and icon-based symbols to program a "bee" to navigate through a digital maze. Project Bloks is made up of physical, modular blocks and interchangeable pucks (Figure 1), which represent instructions for directions (top, down, left, right) or icon-based commands (get flower, make honey). The dials on the direction pucks can be turned, so that any puck can represent any chosen direction at a given time. In addition, the repeat functionality is expressed with open and close blocks that resemble opening and closing brackets. Blocks placed in between the open and close blocks are repeated according to the numeric condition (between 2 and 9) set on dial of the close block. After the sequence is constructed from left to right starting at the green "go" brick, the green button can be pressed to run the program. The digital bee avatar executes each instruction one by one, until it completes the maze or runs into an obstacle (i.e. hits a wall or tries to collect a flower in a non-flower state). The Bloks currently supports 15 different mazes, covering computational thinking concepts such as conditionals and functions, in addition to loops.

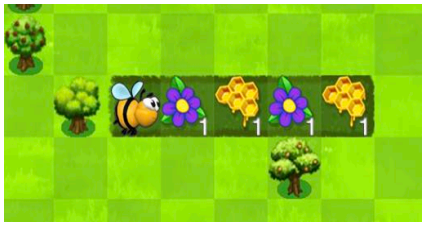
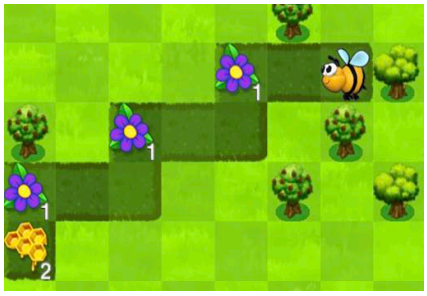
Figure 1. Project Bloks (white blocks, green pucks) and a coding maze on the computer screen.

Procedure

Each session began with a discussion about robots and building activities. Children were then given a brief introduction to Project Bloks before solving a series of 11 puzzles with limited guidance from the researcher. Puzzles were presented in order of increasing difficulty; at Puzzle 8, the "repeat" blocks are introduced. In Puzzles 8 and 9, the repeat bricks are only used to enclose one direction brick; in puzzles 10 and 11, the increased complexity requires up to 4 blocks to be enclosed in the repeat bricks. Because these latter puzzles require children to think in a more sophisticated way, these are the ones that I chose to analyze.

One of the design principles of Project Bloks, which was also a consequence of its physicality, was that the limited number of physical blocks would eventually prompt children to consider solutions to save blocks, especially in the advanced puzzles. Thus, Table 1 displays both the extended sequence (as if there were infinite blocks) and the Bloks solution, which uses only the available blocks. After the puzzles, children were asked to reflect on their experience using a smiley-o-meter and a few additional questions.

Table 1: Images and solutions for Puzzles 10 and 11

	Image	**Extended sequence**	**Bloks solution**
Puzzle 10		right, get flower, right, make honey, right, get flower, right, make honey	repeat x 2 [right, get flower, right, make honey]
Puzzle 11		left, left, get flower, down, left, left, get flower, down, left, left, get flower, down, make honey, make honey	repeat x 3 [left, left, get flower, down], make honey, make honey

DATA AND FINDINGS

The sessions were video-recorded and transcribed, and I captured all intermediary versions of the code at runtime, when children pressed the green (run) button. All 11 children who attempted Puzzle 10 successfully completed it, with an average of 4 runs (range: 1 to 9) and an average duration of 4m28s (range: 1m43s to 8m54s). For Puzzle 11, all 7 children who attempted it were

successful, with an average of 4 runs (range: 1 to 8) and an average duration of 3m34s (range: 1m55s to 6m05s). Findings are presented below with pseudonyms.

Q1: Are the affordances of Project Bloks conducive to the perception of need and learning loops?

By design, Bloks has a limited number of blocks; this makes it impossible to physically represent the extended sequence from Table 1, thereby revealing an opportunity for children to engage with loops. This is a departure from on-screen programming languages, in which the number of available blocks are infinite; the need for loops might appear much later, if at all. In Puzzles 10 and 11, I observed that indeed the limited-block design prompted children to think harder. 6 of 11 children in Puzzle 10 and 2 of 7 children in Puzzle 11 made explicit verbal utterances about needing more blocks, such as "*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. For other children, the limited-block design directly triggered the need for a repeat, with utterances like "*I can just repeat that*" or "*Do you do repeat here?*" One child, Haruki, excitedly exclaimed, "*Ah, yes!*" upon realizing that he needed to add repeat blocks to his program.

I was also interested in validating the physical design of the "repeat" block, which resembled the shape of opening and closing brackets. Children properly oriented and positioned the two blocks, and because of the brackets, the sequence to be repeated was made apparent by their physical encompassment. Three children used both hands simultaneously on the open and close blocks, mimicking the bracket shape with their hands, and in doing so, enacting the idea of an enclosed block of commands with their bodies as a metaphor of a loop.

For both Puzzles 10 and 11, all children who attempted the puzzles were eventually able to complete them. Three children consistently and successfully applied the *planning* strategy for problem-solving (Fessakis et al., 2013), completing both puzzles on their first run or on their second run with minor changes. An exemplary case of this is Mallory on Puzzle 10.

Mallory begins by placing blocks for the full sequence: right, get flower, right, make honey, right (Figure 2a). Upon realizing that there are not enough blocks to continue this way, she stops and asks, "*Does the repeat block do like… does the repeat block repeat like two different ones [blocks]? Or just one?*" Her inquiry demonstrates that the limited-block design did trigger the need for a different type of control flow structure. She initially places the open and close repeat blocks around the first two blocks (Figure 2b), and then 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 (Figure 2c) and says, "*Wait, what if I do that?*" while using both hands to complete the repeat blocks. The simultaneous action of both hands demonstrates the affordance of the bracket design and how it mapped to Mallory's gestures. She then points at the maze, counting the squares as "*1, 2, 1, 2*" to determine the number of times to repeat. She runs the program successfully. Mallory's case suggests 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.

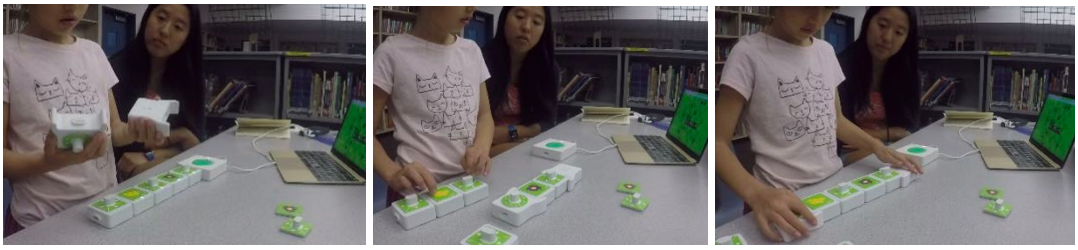


Figure 2a, 2b, 2c. Mallory solving Puzzle 10.

Q2: What kind of difficulties do children encounter when learning about loops?

Although all children eventually solved Puzzles 10 and 11, the variation in the number of programs ran and duration can be explained by a few difficulties. First, in the puzzles before Puzzle 10, the "repeat" block only repeated a single command; Puzzle 10 required a sequence of 4 commands to be repeated. Almost half of the children (5 of 11) were unsure about whether or not they could use the repeat with more than one command.

Second, in Puzzle 10, I observed that 6 children ran programs that included the exact sequence: right, get flower, make honey, get flower, make honey. While this correctly matches the visual representation of the Puzzle 10, there appeared to be a hurdle in understanding the relationship between the symbolic representation and the corresponding action. The first command is to move "right," which demonstrates that the children were initially thinking in terms of action, until they saw the honeycomb in the next square. The evident confusion of mapping representation to action warrants further work.

Finally, in Puzzle 11, 4 of the 7 children also ran the program without adjusting the dial on the close block, suggesting that they may not have considered both components of the repeat block together (i.e. determining what to repeat and how many times to repeat it).

DISCUSSION

These studies, and the preliminary analysis of Puzzles 10 and 11, 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. In those instances, children can create long programs without using loops. Project Bloks, however, 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. This is a contribution to research in tangible interfaces for programming, and possibly a design principle that could be explicitly used by researchers to design similar systems – even with more complex CT concepts.

The studies also illuminated some difficulties that children encountered with learning loops. Although we often assume that children should progress from the simple to the complex, this research 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 more than one command could be repeated. In other words, in my original curriculum, children would first repeat a single command (in Puzzles 8 and 9) and later repeat several commands (Puzzle 10 and 11). I found that almost half of the children were unsure of whether the repeat block would also work for multiple blocks, after having used it for one block. This finding suggests that when introducing a programming 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. I hypothesize that introducing loops with multiple blocks might be more productive, and that this principle of "complex first" might also apply to other constructs, such as conditionals and functions.

The misalignment of the figural and the functional also warrants further research to understand why that happens. This can inform changes in both the physical design of programming interfaces and the facilitation methods for programming activities. In addition, I observed preliminary evidence of distributed cognition; several children focused on determining what to repeat before considering how many times to repeat it. A closer examination of this process will be important for the future design of repeat blocks in tangible and visual programming languages.

Given my narrow focus in this study on how children learn loops, I acknowledge that these results are promising but also preliminary. My findings and larger dataset reveal patterns that may be applicable to other computational thinking concepts, such as functions and conditionals. In future work, I also seek to conduct similar studies with children of other backgrounds and cultures, in order to determine how tangible programming platforms can be equitably designed to help all children learn.

ACKNOWLEDGEMENTS

I would like to thank the Amir Lopatin Fellowship and the Lopatin family for this generous award; the participants of this research for their time and willingness to share; and Paulo Blikstein and the Transformative Learning Technologies Lab at Stanford Graduate School of Education for their support and feedback.

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.

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

Chawla, K., Chiou, M., Sandes, A., Blikstein, P. (2013). Dr. Wagon: A 'stretchable' toolkit for tangible computer programming. ACM IDC 2013.

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.

Horn, M., Jacob, R. (2007). Tangible Programming in the classroom with Tern. ACM CHI 2007.

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.

Lee, Y. (2010). Developing computer programming concepts and skills via technology-enriched language-art projects: A case study. Journal of Educational Multimedia and Hypermedia, 19(3), 307-326.

Minogue, J., & Jones, M. (2006). Haptics in education: Exploring an untapped Sensory Modality. *Review of Educational Research*.

O'Malley, C., & Fraser, D.S. (2004). Literature review in learning in tangible technologies. *NESTA futurelab report 12*, Bristol.

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.

Sullivan, A., Bers., M.U., Mihm, C. (2017). Imagining, playing, & coding with KIBO: Using KIBO robotics to foster computational thinking in young children. In the proceedings of the International Conference on Computational Thinking Education 2017.

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

Xu, D. (2005). Tangible user interface for children - An overview.