Combining Non-Programming Activities With Programming for Introducing Foundational Computing Concepts

Shuchi Grover, Stanford University, shuchig@cs.stanford.edu
Nicholas Jackiw, SRI International, nicholas.jackiw@sri.com
Patrik Lundh, SRI International, patrik.lundh@sri.com
Satabdi Basu, SRI International, satabdi.basu@sri.com

Abstract: This paper describes the design of non-programming activities aimed at aiding early exploration of hard-to-learn introductory CS concepts (specifically, variables and loops) as well as accompanying bridging programming activities in Scratch. These interactive digital and unplugged activities draw on recent research in dynamic math representations. Empirical research that examines the use of these activities in diverse classrooms shows promise of our unique approach and also points to improvements for future iterations of this design research.

Motivation

Computing is transforming innovation in every discipline through becoming an integral tool that is spurring new ways of doing and thinking. This reality has pushed a rethink of learning in this digital age. "Computational Thinking" (CT) and computer science (CS) skills are now acknowledged as foundational competencies for every child (Grover & Pea, 2018). CS and CT education is scaling in K-12 classrooms in the US and internationally.

Learning to program is a central feature of introductory CS curricula in K-12 classrooms. Scholarly literature from the last three decades documents the difficulties that learners have with deeper conceptual learning of specific concepts integral to programming as well as computing more broadly (e.g. du Boulay, 1986). Recent research conducted in the context of popular block-based programming environments using for teaching CT and introductory CS document that difficulties still persist in the learning of concepts such as variables and loops (Grover et al, 2015). They also suggest that prior mathematics preparation can be a barrier to successful learning of CS (Grover et al, 2016; Lewis & Shah, 2012). We believe that to achieve "CS for All" novel, engaging approaches to introducing learners to these difficult-to-grasp concepts should be attempted and empirically studied in order to explore alternate pedagogical paradigms instead of the popular (but not always successful) "dive-into-programming-right-away" approach. This current work attempts to achieve this goal by drawing on ideas from past learning sciences and math education research to explore new ways of engaging learners in introductory computing concepts specifically, variables and loops.

Theoretical framework

Trouble understanding variables pervades students' efforts to use expressions and loops to solve problems in introductory programming (du Boulay, 1989). Middle schoolers working in Scratch have trouble conceptualizing variables and how to use them (especially with loops) and their initial perceptions of variables in programming are influenced by the idea of "unknown thing" as in math (Grover, Pea, & Cooper, 2015; Grover & Basu, 2017).

We are inspired by the powerful idea of epistemological pluralism—multiple ways of knowing and thinking (Turkle & Papert, 1990) for the design of *non-programming* digital interactives to aid understanding of programming concepts. We also draw on research in math education that has established the important role of dynamic representations in supporting students' development of conceptual understanding (Drijvers et al., 2009), specifically, innovations in interactive dynamic representations in geometry (Jackiw, 1991). The key insight from that work is that an interactive system can show change over time—often continuously and in response to learner-directed inquiry—to all the values that can instantiate that model. Learning environments also need to account for both individual construction of knowledge and the sociocultural processes that students partake in, in the Vygotskian framing of learning contexts (Cobb (1994). Additionally, constructivist approaches to learning point to providing learners an opportunity to actively construct knowledge by using their intuition and prior knowledge and refining their micro-theories as they interact with artifacts *before being provided explanations* (Bransford & Schwartz, 1999; Schneider et al., 2015). However, open exploration, especially in the context of programming, often leaves novice learners lost and confused (Mayer, 2004). Exploration thus needs to be *designed* to provide appropriate levels of constraints and guidance. We also strive to explicitly bridge non-programming activities and subsequent introductory programming activities for mediating transfer (Engle, 2012; Grover et al., 2014).

The following sections describe the features of one unplugged and 2 digital (non-programming) activities aimed at developing students' early understanding of variables and loops, followed by a brief account of empirical investigations in three public middle school classrooms that used these activities as part of their introductory computer science course. (Space constraints preclude detailed treatment of the designs).

Methods

This section describes the design-based research (DBR) around the design of two digital and one unplugged non-programming activities. These activities were designed as part of a larger effort aimed at exploring novel ways of engaging students in introductory programming concepts, such as variables (along with loops), that novice learners find difficult to grasp. The activities were designed and refined with inputs from teachers (in a participatory design model) and middle-school students (through 2 rounds of piloting involving "thinkalouds").

Design of curricular activities

Designed as a short and preliminary exploration and introduction to key ideas of foundational *concepts* (rather than introductory programming *constructs)* these activities don't attempt to deliver *comprehensive* treatments and are designed for one (or two) class periods. All digital activities begin with students exploring the basic phenomenology of the microworld. They are designed as paired exploration along with whole-class discussions.

'Story Variables' (unplugged) and 'Cats & Ladders' (digital) activities

In *Story Variables* students work collaboratively in pairs to investigate a series of short "stories" all containing *quantities that vary*. For example, *"Excuse me—last week I bought one of these pens here for $1.50. Are you really telling me they now cost $3?*"; "*I watched the basketball game last night. At halftime we were tied, but in the end, they beat us 94-90.*" Through discussions, students come up with a definition of **'variable'**, practice identifying and naming variables meaningfully, and analyze a variable's changing values to determine its specific types and expected ranges. Students identify their own real-world scenarios that involve "variables". As a final activity, they watch a video clip of Pacman and list the different variables they observe. In the *Cats and Ladders* digital activity (Fig 1), we leverage the popularity of cats! Students rescue distraught cats from the upper floors of various buildings by determining the length of the ladder required to reach them from the ground. The activity is divided into multiple stages, each of which is "unlocked" as the student proceeds through the activity. Learners discuss appropriate names for variables (e.g. "height" or "LadderHeight" is not sufficiently discriminating for the 2 ladders). They also discuss the range of possible values (which are often determined by context), and that different variables may naturally reference different (data) types. Finally, they engage in abstraction through a preliminary exploration of arithmetic expressions and that new variables can be synthesized from existing ones.

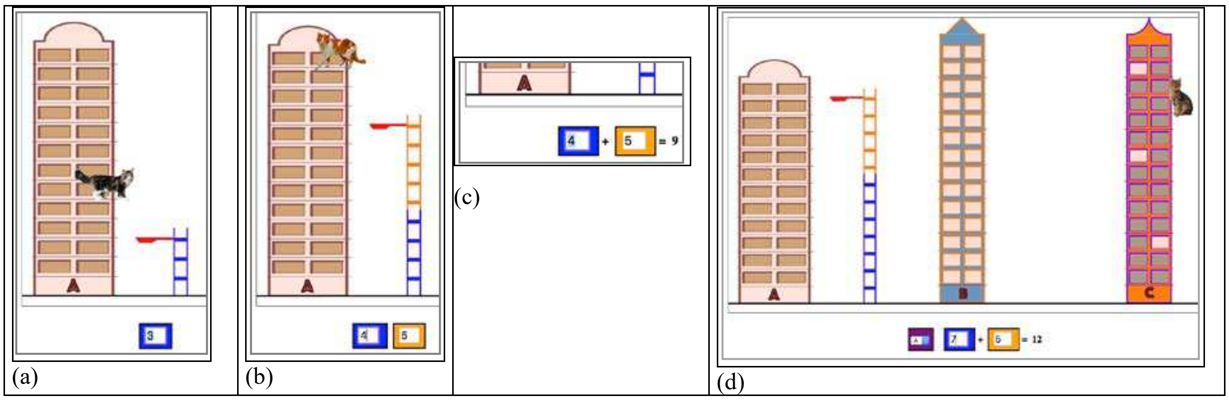
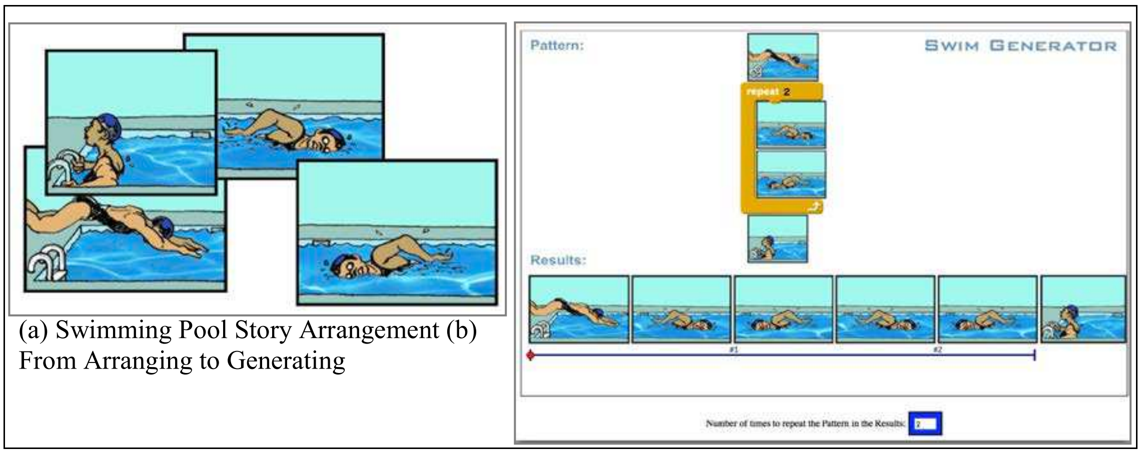


Figure 1. Different stages of the *Cats and Ladders* digital activity.

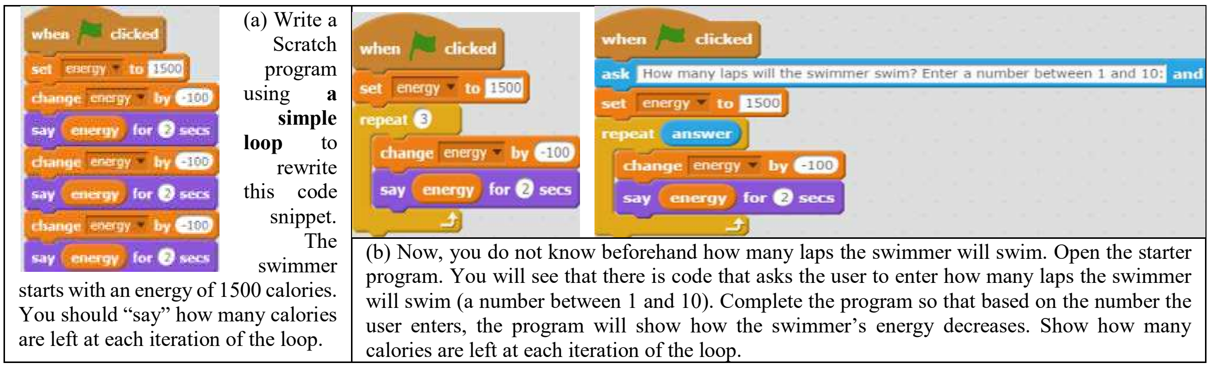
'Graphical Looping' digital activity and bridging Scratch activities

The *Graphical Looping* activity sequence (Fig. 2) introduces students to iterated repetitions of a block of actions within a sequence of events, and develops the idea that we efficiently express such a flow of events in terms of a more *compact specification* of that repetition. Students engage with the idea of action sequences that occur before and/or after a repeating chunk of actions. *Graphical Looping* makes productive use of comic panels as a proxy for source code in a pre-programming context. Comic strips are *atextual* (and thus don't disadvantage ELL students) and contain a formal, and block-structured, grammar for describing action sequences familiar to students. First, students arrange comic panels to tell a "logical story." They proceed to think about how panels to the story could describe longer swims and identify the "inner story" or "repeating unit". They discuss how the total length swum increases and the swimmer's energy decreases with each lap. ***These ideas are revisited in Scratch programming***. The Scratch activities related to loops with variables make explicit connections to *Graphical Looping*. Learners are introduced to the idea of using a "Repeat" block to swim 3 laps, change the (swimmer's) *Energy* value, and also "watch" this value decrease (using the 'Say' block) with each lap (or iteration of the loop). They then move to the idea of a generalized solution where the number of laps is based on an input from the user that is then used as a variable in the Repeat block (Figure 3).



(a) Swimming Pool Story Arrangement (b) From Arranging to Generating

Figure 2. Screenshots of the *Graphical Looping* digital activity.



```
when [flag] clicked
set energy to 1500
change energy by -100
say energy for 2 secs
change energy by -100
say energy for 2 secs
change energy by -100
say energy for 2 secs
```

(a) Write a Scratch program using **a simple loop** to rewrite this code snippet. The swimmer starts with an energy of 1500 calories. You should "say" how many calories are left at each iteration of the loop.

```
when [flag] clicked
set energy to 1500
repeat 3
    change energy by -100
    say energy for 2 secs
```

```
when [flag] clicked
ask How many laps will the swimmer swim? Enter a number between 1 and 10: and
set energy to 1500
repeat answer
    change energy by -100
    say energy for 2 secs
```

(b) Now, you do not know beforehand how many laps the swimmer will swim. Open the starter program. You will see that there is code that asks the user to enter how many laps the swimmer will swim (a number between 1 and 10). Complete the program so that based on the number the user enters, the program will show how the swimmer's energy decreases. Show how many calories are left at each iteration of the loop.

Figure 3. Scratch activities using loops and variables that bridge to *Graphical Looping*.

Study and data measures

These activities were embedded as a 3-to-4 week curricular intervention (length varied by teacher) in a 3 middle school classrooms (N=72; Gr. 6: 17 male, 10 female; Gr. 7: 15 male, 16 female; Gr. 8: 11 male, 3 female) that mirrored the diversity in a large, diverse, urban school district in the U.S. The curriculum was part of an introductory CS course that used Scratch. Three teachers participated in 20 hours of professional development before implementing the curriculum. We conducted mixed method research to: 1) Understand the affordances and design modification needs of these designed activities; and 2) Document and analyze the classroom activity system necessary to support students' productive engagement with these computing concepts.

Our data measures included (a) *Pre-post & formative assessments* that were designed using a robust assessment design framework, and included multiple-choice and open response question types and either used snippets of Scratch code, or real-world (narrative) scenarios; (b) *Post-survey* on students' reactions to the activities; (c) *Interviews & Think-alouds* with 12 students (4 per grade) and 3 teachers; (d) *Final (open-ended) projects in Scratch*: from the 3 classrooms, done (individually or in pairs) analyzed using an elaborate rubric (Grover, Basu, & Schank, 2018). We also analyzed final Scratch projects from ~80 middle school randomly selected from students across the district who were not part of this study (as a comparison).

Results and discussion

Preliminary results (from ongoing analyses) show that all students showed significant gains on the pre-post assessment (Table 1). Preliminary analyses of Scratch projects suggest that students in the sample demonstrated better facility with variables & loops, and their programs were more complex as compared to the comparison projects. Students engaged well with these activities, demonstrating promise of the focus on dynamic representations and relevant examples and contexts. However, the range of students' readiness to engage and learn and their instructional needs varied considerably both within and across the three classrooms. The 6th graders had the most challenges, whereas the high achieving 7th grade class was able to engage most with the conceptual ideas. This was reinforced from students' post-survey feedback on these activities. Overall, students found *Cats*

& Ladders to be most engaging and fun (" *I liked cats and ladders because it was like a real game*"; "*It's… to show us about height. And like the variables change. Because like you could change the numbers...It shows us variables. Like how high the ladder could go, like how we could change it."; It was fun and cool"). Graphical Looping* activity was found to be lacking in challenge by some "*It was too easy*"), however students readily made real-world connections ("*It teaches you the purpose of what loops do...It reminded like if we go to a grocery store and we could come back and get more food from there and then come back..Like each day like you like.. it can be from like school, you go back home. And from the home you go back to school*") We take these results as constructive feedback to improve upon the designs in the next iteration of this DBR.

Table 1. Student pre-post assessment results

	pre Mean	post Mean	p-value	Cohen's *d*
Grade 6 (n=27)	39.2	52.5	p=0.0007	0.65
Grade 7 (n=31)	65.5	80.0	p<.0001	1.02
Grade 8 (n=14)	56.3	68.5	p=0.0009	0.67

Conclusion and future work

"CS for All" in K-12 requires that curricula be designed so all learners succeed in deeper learning of computing regardless of prior academic preparation. This current work attempts to achieve that by exploiting the synergies between key concepts in middle school math and programming, and draw on math education research on the use of dynamic representations to create and examine the use of game-like digital interactives and microworlds that provide early engagement with hard-to-learn computing ideas. While early results are encouraging, they also point toward directions for improvement in future iterations of this design research.

References

Bransford, J. D., & Schwartz, D. L. (1999). Rethinking transfer: A simple proposal with multiple implications. In A. Iran-Nejad & P. D. Pearson (Eds.), *Review of Research in Education,* 24 (pp. 61–101).

Drijvers, P., Kieran, C., Mariotti, M. A., et al. (2010). Integrating technology into mathematics education: Theoretical perspectives. In C. Hoyles , & J. P. Lagrange (Ed.), *Mathematics education and technology-rethinking the terrain* (pp. 89-132). New York, NY: Springer.

duBoulay, B. 1986. Some difficulties of learning to program. *Journal of Ed Computing Research, 2*(1), 57-73.

Engle, R. A., Lam, D. P., Meyer, X. S., & Nix, S. E. (2012). How does expansive framing promote transfer? Several proposed explanations and a research agenda for investigating them. *Ed Psychologist, 47*(3).

Grover, S., Pea, R., & Cooper, S. (2014). Expansive framing and preparation for future learning in middle-school computer science. In *International Conference of the Learning Sciences (ICLS) Conference, 2014*. ISLS.

Grover, S., Pea, R., Cooper, S. (2015). Designing for Deeper Learning in a Blended Computer Science Course for Middle School Students. *Computer Science Education, 25*(2), 199-237.

Grover, S., Pea, R. & Cooper, S. (2016). Factors Influencing Computer Science Learning in Middle School. In *Proceedings of the 47th ACM Technical Symposium on CS Ed. (SIGCSE '16).* ACM.

Grover, S. & Basu, S. (2017). Measuring Student Learning in Introductory Block-Based Programming: Examining Misconceptions of Loops, Variables, and Boolean Logic. In Proceedings of *SIGCSE '17.*

Grover, S. Basu, S. & Schank, P. (2018). What We Can Learn About Student Learning From Open-Ended Programming Projects in Middle School Computer Science. In Proceedings of *SIGCSE '18.* ACM.

Jackiw, N. (1991, 2009). *The Geometer's Sketchpad* (computer software V1, V5). Key Curriculum Press.

Lewis, C. M., & Shah, N. (2012). Building upon and enriching grade four mathematics standards with programming curriculum. In *Proceedings of the 43rd Technical Symposium on CS Education.* ACM.

Mayer, R. E. (2004). Should there be a three-strikes rule against pure discovery learning?. *American Psychologist*, *59*(1), 14.

Samurcay, R. 1989. The concept of variable in programming: Its meaning and use in problem-solving by novice programmers. In E. Soloway and J. C. Spohrer, editors, *Studying the Novice Programmer*, 161–178.

Schneider, B., Wallace, J., Blikstein, P., & Pea, R. (2013). Preparing for future learning with a tangible user interface: the case of neuroscience. *IEEE Transactions on Learning Technologies*, *6*(2), 117-129.

Turkle, S. & Papert, S. (1990). Epistemological pluralism and the reevaluation of the concrete. SIGNS: Journal of Women in Culture and Society, 16(1), 128—157.

Acknowledgements

We thank NSF(#1543062) for grant support; Bryan Twarek, Matthias Hauswirth, Jeremy Roschelle, Yesica Lopez, and our district partner, teachers & students for contributions to this research.