Defining, Designing, and Documenting Computational Thinking Across K-12 Education

David Weintrop, University of Maryland, weintrop@umd.edu

Abstract: Smartphones, tablets, and laptops are becoming the lenses through which we see, interpret, and interact with the world. As such, for young learners growing up in this technological landscape, being able to recognize the capabilities and limitations of these technologies, think critically about the roles they play in society, and most crucially, to be able to meaningfully participate in this technological culture is essential. Much of the current effort to prepare learners for this future is collected under the umbrella label Computational Thinking, an intentionally vague term comprised of skills and concepts that enable computational inquiry across diverse subject areas and support new forms of creative expression. As part of the effort to bring computational thinking to all, new programming environments, learning platforms, and educational initiatives are being developed and deployed. This paper presents my prior, current, and intended future work towards better understanding the nature of computational thinking and ways to bring it to all learners.

Introduction

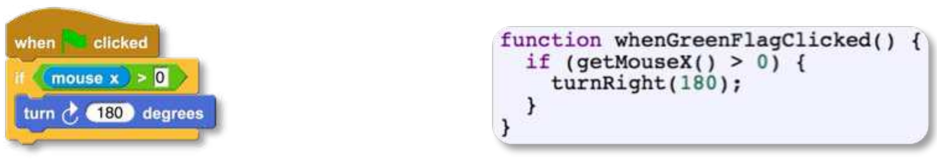
Computation is changing our world. In response to the growing need for all students to develop technical and computational literacy skills, new programming environments, learning platforms, and educational initiatives are being developed and deployed to prepare learners for their digital futures. Computational thinking, and its constituent skills, has far reaching applicability, supporting inquiry across diverse subject areas, enabling new forms of expression and creative outlets, and opening the doors to new opportunities for learners in classrooms and beyond. While there is a great deal of activity towards broadly engaging students with computational thinking, many open and consequential questions about how best to accomplish these goals remain.

Broadly, my research seeks to understand how best to support learners in developing meaningful understandings of computational ideas and positive attitudes towards computing. More specifically, I study the design of computational learning environments, looking at the interplay between the representations used, the nature of the learning activity, and the context in which the experience is situated, to make sense of how these aspects collectively shape learners' emerging understanding of powerful ideas in computing. I am particularly interested in understanding how specific features of educational technologies and computational tools make ideas more intuitive and accessible and using these findings to inform the design of new learning experiences. My work lies at the intersection of the Learning Sciences, human-computer interaction, and computational thinking education, with an emphasis on the application of theory to the design and evaluation of innovative computational learning environments. While much of my work focuses on learners developing computational literacy skills, my work extends to using computational environments to deepen learning across an array of disciplines. My research contributes to our understanding of how best to engage people in reasoning about powerful computational ideas and support their ability to develop and apply computational thinking practices. A second contribution of my work is advancing the design of environments and technologies that can broaden participation in computing, bringing learners from diverse and historically underrepresented backgrounds into computing. To date, my research has focused on two main areas: the design of introductory programming environments and the integration of computational thinking into existing subjects as a way to broaden participation in computing.

Designing, understanding, and evaluating introductory programming environments

In my dissertation, I pursued questions on the relationship between the design of introductory programming environments and the knowledge and practices they promote in computer science classrooms. Specifically, I studied the impact of introducing learners to programming through graphical, block-based programming environments (Fig. 1A) such as *Scratch* (Resnick et al., 2009) and *Alice* (Cooper, Dann, & Pausch, 2000), compared to conventional text-based programming languages.

Block-based programming is increasingly becoming the way the learners are being introduced to programming and big ideas in computational thinking more broadly. A growing body of work is finding that the block-based approach is effective at engaging diverse audiences in computing and lowering the barrier to programming (Kelleher, Pausch, & Kiesler, 2007; Maloney, et al., 2008; Ryoo, Margolis, Lee, Sandoval, & Goode, 2013).

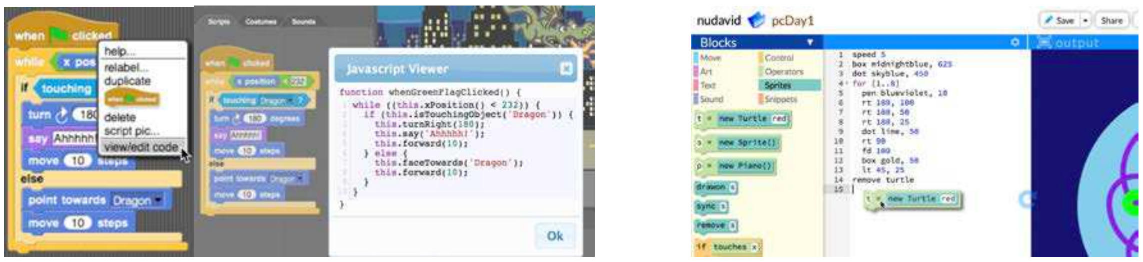


```
function whenGreenFlagClicked() {
  if (getMouseX() > 0) {
    turnRight(180);
  }
}
```

(A) (B)

Figure 1. Comparable block-based (A) and text-based (B) programs.

This is an essential first step in identifying the utility of block-based tools, but follow-up questions looking at how the modality influences emerging understandings and practices had yet to be answered in a systematic way before my dissertation work. In the Learning Sciences, similar questions have been pursued looking at the role of representation in supporting and shaping understanding in mathematics and science domains (Kaput, Noss, & Hoyles, 2002; Sherin, 2001; Wilensky & Papert, 2010). In bringing this lens to the design of introductory programming environments, my goal is to advance our understanding of how and why specific features work and to use those findings to inform the design of the next generation of computational learning environments. The questions I pursued in my dissertation ask how the block-based modality affects learners' emerging understandings of core programming concepts and if and how concepts learned and practices developed in block-based tools carry over to more conventional text-based programming languages. To answer these questions, I designed a quasi-experimental, mixed-methods study across three high-school computer science classrooms, with each class following the same curriculum, but using a different programming environment (Weintrop, 2016). I created both the curriculum and the programming environments used in the study. I conducted two iterations of this design-based research study in successive school years in an urban, public high school, using findings from year one to inform the design of the programming environments and curriculum for the second. Both iterations included the design and implementation of three modes of two different introductory programming environments that differentially support block-based, text-based, and hybrid block/text approaches to programming (Fig. 2). The environments I created were based on Snap! (Harvey & Mönig, 2010) in year 1 (Fig. 2A) and Pencil Code (Bau et al., 2015) in year 1 (Fig. 2B).



```
function whenGreenFlagClicked() {
  while ((this.xPosition() < 232)) {
    if (this.isTouchingObject('Dragon')) {
      this.turnRight(180);
      this.say('Ahhhhh!');
      this.forward(10);
    } else {
      this.faceTowards('Dragon');
      this.forward(10);
    }
  }
}
```

```
speed 5
box midnightblue, 625
dot skyblue, 450
for [1..8]
  pen blueviolet, 10
  rt 180, 100
  rt 180, 50
  rt 180, 25
  dot lime, 50
  rt 90
  fd 100
  box gold, 50
  lt 45, 25
remove turtle
```

(A) (B)

Figure 2. Two iterations of hybrid blocks/text environments from my dissertation study.

My dissertation shows that modality matters. One contribution of this work is the finding that students working in the block-based form outperformed their peers who used an isomorphic text-based versions (Weintrop & Wilensky, 2017b). Further, analyses of students answering questions on central computing concepts in either block-based or text-based modalities show that students' ability to answer questions differed by modality, with student performing better in block-based contexts across a number of implementations and languages (Weintrop, Killen, & Franke, 2018; Weintrop & Wilensky, 2015). In following students as they moved from introductory environments to a professional programming language (Java), we found the differential learning outcomes that occurred in the introductory tools did not better prepare learners for later computer science instruction (Weintrop & Wilensky, Under Review). This presents a new set of research questions about how to support learners in transitioning from introductory to professional tools, a line of work I intend on pursuing. A second contribution of this work is the creation and evaluation of two hybrid programming environments (Figure 2) that blend features of block-based and text-based environment (Weintrop & Wilensky, 2017a).

Broadening participation by integrating computing across the curriculum

The second focus of my research is on exploring ways to broadening participation in computing through the design of accessible computational learning experiences that are easily implemented in existing classrooms. In particular, I am looking at ways to integrate computational thinking into existing mathematics and science classrooms (Pei, Weintrop, & Wilensky, 2018; Weintrop et al., 2016). The motivation for this approach is threefold: (1) it builds on the symbiotic relationship for learning between computational thinking and mathematics and science domains, (2) it addresses practical concerns of reaching all students and having proficient teachers, and (3) it brings science and mathematics education more in line with contemporary professional practices in these disciplines. By integrating computational thinking into existing classrooms, this work contributes a novel, easily adoptable, and scalable strategy for engaging diverse populations in computing that is theoretically grounded, practically feasible, and fits within existing educational infrastructure. To date, the focus has been on high school mathematics and science classrooms and focused on learners and curricular materials. I am currently in the process of expanding the scope of this integrative approach to include English Language Arts and other humanities and artistic subjects. Additionally, I am working on a project studying pre-service teachers to understand what existing knowledge resources they possess that can be used to help them integrate computational thinking into their teaching. In doing so, the hope is to develop approaches to help teachers draw on their prior experiences with computing and technology to advance the goal of bringing computational thinking instruction to all learners.

References

Bau, D., Bau, D. A., Dawson, M., & Pickens, C. S. (2015). Pencil Code: Block Code for a Text World. In *Proceedings of the 14th International IDC Conference* (pp. 445–448). New York, NY, USA: ACM.

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges*, *15*(5), 107–116.

Harvey, B., & Mönig, J. (2010). Bringing "no ceiling" to Scratch: Can one language serve kids and computer scientists? In J. Clayson & I. Kalas (Eds.), *Proc. of Constructionism 2010* (pp. 1–10). Paris, France.

Kaput, J., Noss, R., & Hoyles, C. (2002). Developing new notations for a learnable mathematics in the computational era. *Handbook of International Research in Mathematics Education*, 51–75.

Kelleher, C., Pausch, R., & Kiesler, S. (2007). Storytelling alice motivates middle school girls to learn computer programming. In *Proc. of the SIGCHI 2007* (pp. 1455–1464).

Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: Urban youth learning programming with Scratch. *ACM SIGCSE Bulletin*, *40*(1), 367–371.

Pei, C. (Yu), Weintrop, D., & Wilensky, U. (2018). Cultivating Computational Thinking Practices and Mathematical Habits of Mind in Lattice Land. *Mathematical Thinking and Learning*, *20*(1), 75–89.

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

Ryoo, J. J., Margolis, J., Lee, C. H., Sandoval, C. D., & Goode, J. (2013). Democratizing computer science knowledge: transforming the face of computer science through public high school education. *Learning, Media and Technology*, *38*(2), 161–181.

Sherin, B. L. (2001). A comparison of programming languages and algebraic notation as expressive languages for physics. *International Journal of Computers for Mathematical Learning*, *6*(1), 1–61.

Weintrop, D. (2016). *Modality Matters: Understanding the Effects of Programming Language Representation in High School Computer Science Classrooms* (Dissertation). Northwestern University, Evanston, IL.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, *25*(1), 127–147.

Weintrop, D., Killen, H., & Franke, B. (2018). Blocks or Text? How programming language modality makes a difference in assessing underrepresented populations. In *Proc. of International Conference on the Learning Sciences 2018*. London, UK.

Weintrop, D., & Wilensky, U. (2015). Using Commutative Assessments to Compare Conceptual Understanding in Blocks-based and Text-based Programs. In *Proc. of the Eleventh Annual International Conference on International Computing Education Research* (pp. 101–110). New York, NY, USA: ACM.

Weintrop, D., & Wilensky, U. (2017a). Between a Block and a Typeface: Designing and Evaluating Hybrid Programming Environments. In *Proc. of the 2017 Conference on Interaction Design and Children* (pp. 183–192). New York, NY, USA: ACM.

Weintrop, D., & Wilensky, U. (2017b). Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms. *ACM Transactions on Computing Education (TOCE)*, *18*(1), 3.

Weintrop, D. & Wilensky, U. (Under Review). How Block-based, Text-based, and Hybrid Block/Text Modalities Shape Novice Programming Practices.

Wilensky, U., & Papert, S. (2010). Restructurations: Reformulating knowledge disciplines through new representational forms. In J. Clayson & I. Kallas (Eds.), *Proc. of Constructionism 2010*. Paris, France.