Teachers College Record
The Voice of Scholarship in Education
Book Reviews

Connected Code: Why Children Need to Learn Programming

reviewed by Peter Rich & Brian Lindley Jones

Title: Connected Code: Why Children Need to Learn Programming
Author(s): Yasmin B. Kafai & Quinn Burke
Publisher: MIT Press, Cambridge
ISBN: 0262027755, **Pages:** 200, **Year:** 2014
Search for book at Amazon.com

The 3 Rs—reading, writing, and arithmetic—have long been promoted as the core skills students need to learn in order to competently succeed in all other subject areas. Programming is increasingly being touted as the 4th 'R' for a 21st century education. In their book, *Connected Code*: *Why Children Need to Learn Programming,* Yasmin B. Kafai and Quinn Burke draw from their own extensive experience teaching children to code. They argue that it is not simply enough for students to learn to code, but rather for all pupils to become computational participants in today's increasingly digital society. From this perspective, learning to program is to computational participation as writing is to literacy. Computational participation goes beyond programming to include collaboration in a maker society, just as literacy goes beyond the fundamental act of writing. In addition to advocating that everyone should learn to code, *Connected Code* presents the developing idea of computational participation, encouraging more productive, authentic, and creative learning through collaborative processes.

In Chapter One, Kafai and Burke expand on the notion of computational thinking to include the authentic and social aspect of learning to code in what they term as computational participation, an idea that recognizes the increased capacities, social connections, and worth that occurs as a result of learning to code. They identify this social aspect of computing as an essential element for the comeback of coding. While the 1980s saw an investment in Logo-based computing (often to teach geometry and logic), the practice of coding waned as the decade ended. In the mid-1990s, there was a resurgence of interest in teaching children to code through after-school clubs, museum exhibits, and special programs. The authors discuss Mitch Resnick's work and the creation of Scratch, a visual programming language meant to teach children how to code.

Chapters Two through Five continue to lead readers through the journey of Scratch and its evolution as a tool for promoting computational participation. Kafai and Burke suggest that the technical aspects of learning to code are second to the ability to think and plan logically. The authors point out that promoting and teaching logical thinking is far from a novel educational goal. In contrast to early efforts to teach programming as a way of learning mathematics, *Connected Code* argues that children should learn to program for its own sake and, more importantly, for authentic reasons. Authenticity in programming means projects must be: (a) personally relevant, (b) reflect expert thinking in a domain, and (c) extend to real-world and authentic audiences. They share the stories of students who used code as a way to demonstrate what they had learned in mathematics, history, language arts, and other projects that were personally relevant to both the students creating the materials and their audiences.

Kafai and Burke build on the idea that the tools that promote coding must follow the building metaphor offered by Resnick and Silverman (2005). That is—the tool must have low floors (intuitive enough for beginners), high

ceilings (capable enough for expert applications), and wide walls (diverse enough for facilitating a wide range of products). To this, they add that the tool must have windows; that is, the tool must promote the opening of new communities. *Connected Code* tells stories of children who spend hours working together in creating, curating, and modifying each other's code—all without ever having met each other. They promote remixing, what might be viewed as cheating in a traditional classroom setting. Rather than having to begin every project from the ground up, students in the Scratch community share and repurpose code. Remixing code acts as a pathway to better coding practices, comprehending complex concepts, participating in and understanding of social norms of a coding community, and establishing a culture of shareability. Kafai and Burke argue that remixing reflects the philosophy of the open-source movement, and the advancement of better coding practices and products. The authors extensively promote the notion that computational participation is best fostered when work is freely shared and collaborated on.

In Chapter Six, the discussion moves beyond Scratch and demonstrates how computational participation extends to a variety of contexts. The authors present different tools that promote tangible computing, such as MaKey MaKey, LilyPad Arduino, Lego Mindstorms, HyperGami, and Math on a Sphere. From this perspective, computational participation moves beyond the traditional computer software environment into the physical through textiles, toys, and other objects that pervade everyday life. By bringing together physical materials and programming, students of all interests and walks of life can become computationally involved and better understand how they might improve or alter their day-to-day interactions. This chapter is an eye-opening experience on how the intersection of the tangible world with the coding world can foster creative minds and materials.

Chapter Seven brings the discussion back to teaching and traditional education. Kafai and Burke explore ways in which coding might be integrated with current curricula. They present research wherein primary students used Scratch in both after-school and during-school contexts, concluding that in-school contexts lead to greater participation, but less collaboration and creativity. Rather than promote that all coding be done in after-school clubs, the authors explore ways in which teachers might encourage the after-school mentality during in-school integrated projects.

The book concludes in a similar vein as it began. While *Connected Code* acknowledges that the goal of learning to code is not that every child will become a computer scientist, it argues that a basic understanding of coding allows us to understand the systems that undergird and drive a 21st century society. More than that, however, learning to code empowers people and provides resources to solve problems. Perhaps most importantly, coding allows students to express themselves in order to communicate and interact with others.

Connected Code is a book for anyone interested in understanding both the rationale and research behind creating a collaborative maker-oriented approach to primary education. The majority of the book explores this by describing the Scratch programming environment for kids. As I regularly use Scratch to teach children to program, *Connected Code* provides the vision of a community of creators. Kafai and Burke promote that learning to program extends beyond traditional school walls and even the confines of a computer. Educators who are interested in better understanding new possibilities in bringing together technology and engineering in the classroom to promote computing with tactile elements may find the latter chapters a source of inspiration. In sum, *Connected Code* is a book that presents ideas and research that may inspire those looking to understand how and why teaching children to program can lead to a collaborative community of creators.

References

Resnick, M. & Silverman, B. (2005). Some reflections on designing construction kits for kids. In *Proceedings of the 2005 Conference on Interaction Design and Children* (pp. 117-122). New York, NY: ACM.