Chapter 30
From Virtual to Physical Problem Solving in Coding:
A Comparison on Various Multi-Modal Coding Tools for Children Using the Framework of Problem Solving

Kening Zhu
City University of Hong Kong, China

ABSTRACT

Using coding education to promote computational thinking and nurture problem-solving skills in children has become an emerging global trend. However, how different input and output modalities in coding tools affect coding as a problem-solving process remains unclear. Of interest are the advantages and disadvantages of graphical and tangible interfaces for teaching coding to children. We conducted four kids coding workshops to study how different input and output methods in coding affected the problem-solving process and class dynamics. Results revealed that graphical input could keep children focused on problem solving better than tangible input, but it was less provocative for class discussion. Tangible output supported better schema construction and casual reasoning and promoted more active class engagement than graphical output but offered less affordance for analogical comparison among problems. We also derived insights for designing new tools and teaching methods for kids coding.

INTRODUCTION

Educators and researchers have identified the benefits of learning computer programming (a.k.a. coding) at a young age. Papert (1980) envisioned that learning coding could help children overcome fear of math, and help them to learn actively. This insight has been later emphasized and proven by other researchers. Clements et al. found that children who learned programming outperformed those who did not in reflectivity, divergent thinking, and metacognitive abilities (Clements, 1986). Strand et al. (1986) reported that programming facilitated collaboration among students, improved their social skills, and

DOI: 10.4018/978-1-7998-3016-0.ch030

Copyright © 2021, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

encouraged greater focus on their work. As the inventors of Scratch (Maloney et al., 2010) and ScratchJr (Flannery et al., 2013), Resnick et al. (2009) argued that through coding, children can develop "computational thinking" (Wing, 2006), which could cultivate creativity and important problem-solving strategies. Solomon (2005) suggested that "computer programming can be a useful, creative, and thoroughly entertaining second language for students at all levels." More recently, Wong et al. (2015) investigated the impact of coding education at primary schools in Hong Kong, and reported an improvement in the overall performance of the students in mathematics, and the development of soft skills, after learning coding.

The revealed benefits of coding education have motivated many research efforts on new coding tools for children as an alternative to the conventional textual programming environment. The concept of visual programming (Bragg & Driskill, 1994) uses graphical symbols to represent coding concepts and allows children to compose computer programs by piecing graphic icons together. Although textual coding has its own benefits of low viscosity and high expandability (Green & Petre, 1996), visual programming, especially block building, has significant advantages over textual mode for children just entering the venue of computing. It allows children to focus more on learning the computational concepts rather than the complex syntax. As the tangible user interfaces (TUIs) emerged, researchers have also developed tangible block-based programming tools (e.g., Horn & Jacob, 2007). Studies show that graphical-user-interface-based visual programming can achieve better independence in study and learning outcome, while tangible programming is easier to use, more inviting, and more supportive for collaboration (Bers & Horn, 2009; Horn, Solovey, Crouser & Jacob, 2009; Resnick et al. 2009).

In spite of many research efforts on children's programming education, existing work has mostly emphasized comparing user experience (e.g., the usability of coding interface) with different interface modalities. For example, Horn, Crouser and Marina (2012 suggested that children might have difficulty in operating the mouse on a graphical user interface. The emergence of touch surface technology has removed such barriers. Prior studies of hybrid coding interfaces advocated for providing users with the flexibility to choose a preferred modality (Horn, Crouser & Marina, 2012), but the impact of different systems on learning dynamics (e.g., peer interaction and interaction between teachers and students) is largely overlooked. Oviatt el. al's (2012) research revealed the impacts of different interfaces on ideation and problem solving for high school and university students, but how different systems would affect young children in learning computational thinking and problem solving is still unclear. A recent study (Sapounidis, Demetriadis & Stamelos, 2015) compared children's performance with graphical input and tangible input for robotics, and it suggested fewer programming errors occurred and better debugging was achieved with tangible input. This research examined different input methods for one unified output presentation (a physical robot). However, there was no detailed discussion on the potential affordances of different embodiment (graphical/tangible) of the problem itself, which could affect the problem-solving process (Czarnocka, 1995). In other words, there still lacks investigation on the affordance of different combinations of input and output modalities on passing problem-solving-related knowledge to children.

In this paper, we explore whether different variations of input and output combinations of coding tools for children (graphical input + graphical output, graphical input + tangible output, tangible input + graphical output, tangible input + tangible output) can adequately support the process of learning and problem solving. Based on the classification by Brown and Chandrasekaran (2014), we conceptualized the act of computer programing as one kind of *design-problem-solving* process, where the final goal is clearly known but the problem is ill-structured and open-ended with unclear decomposition plans. Such a process requires important cognitive skills, including problem schema construction, analogical com-

16 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/from-virtual-to-physical-problem-solving-in-coding/261049

Related Content

Intrusion Detection System Using Deep Learning
Meeradevi, Pramod Chandrashekhar Sunagarand Anita Kanavalli (2022). *Deep Learning Applications for Cyber-Physical Systems (pp. 160-181).*
www.irma-international.org/chapter/intrusion-detection-system-using-deep-learning/293129

Moving Forward a Parsimonious Model of Eco-Innovation: Results From a Content Analysis
Yudi Fernandoand Wen Xin Wah (2020). *Disruptive Technology: Concepts, Methodologies, Tools, and Applications (pp. 111-124).*
www.irma-international.org/chapter/moving-forward-a-parsimonious-model-of-eco-innovation/231183

Technological Disruption as a Driving Force for Coopetition: The Case of the Self-Driving Car Industry
Rauno Rusko, Lilli Alatalo, Joel Hänninen, Juho Riipi, Ville Salmelaand Joel Vanha (2020). *Disruptive Technology: Concepts, Methodologies, Tools, and Applications (pp. 819-836).*
www.irma-international.org/chapter/technological-disruption-as-a-driving-force-for-coopetition/231220

Software-Defined Networking in Aviation: Prospects, Effectiveness, Challenges
Roman Odarchenko (2019). *Cases on Modern Computer Systems in Aviation (pp. 147-175).*
www.irma-international.org/chapter/software-defined-networking-in-aviation/222187

Petri Net Based Deadlock Prevention Approach for Flexible Manufacturing Systems
Chunfu Zhongand Zhiwu Li (2012). *Computer Engineering: Concepts, Methodologies, Tools and Applications (pp. 445-463).*
www.irma-international.org/chapter/petri-net-based-deadlock-prevention/62458