Horn, M. & Bers, M. (2019). Tangible Computing. In The Cambridge Handbook of Computing Education Research (S.A. Fincher and A.V. Robins, Eds.). Cambridge University Press.

Tangible Computing

Michael Horn, Marina Bers

1. Introduction

Seymour Papert's book, *Mindstorms: Children, Computers, and Powerful Ideas* (1980), featured a startling photograph for the time—a large dome-like robot, a floor turtle, that could be programmed by children to draw geometric forms on paper sheets. This was Logo. Not just the turtle on the screen that spread throughout the world on the floppy disks of the 1980s, but a physical-digital hybrid system, an embodied configuration of metal, glass, plastic, and bits. The spirit of physicality in Logo, and many other educational languages of the time (see Kelleher & Pausch, 2003), became mostly metaphorical—the turtle moved in the physical space of a virtual world of abstracted geometry. But, metaphor though it was, the creators of these languages saw physicality as an essential link between children's embodied experiences in the world and the new universe of computer code.

This chapter is about *tangibility* in Computer Science Education. And, even though the term "tangible" didn't gain widespread use in Human-Computer Interaction until the turn of this century, the history of computing education is clearly anchored in tangible roots that have grown and blossomed over the last thirty years. We see these roots in early educational programming language paradigms. diSessa and Abelson spoke of "spatial metaphor" and "naive realism" in their design of Boxer (diSessa & Abelson, 1986); Papert evoked the concept of "body syntonic" reasoning in children's use of Logo (1980); and, with Karel the Robot (1981), Richard Pattis sought to introduce Computer Science to learners in terms of navigating a grid world. We see other echos of tangibility in the panoply of visual programming languages that rely heavily on physical and spatial metaphors to represent concepts such as encapsulation, scope, flow-of-control, and syntax (Repenning, 1993; Erwin, Cyr, & Rogers, 2000; diSessa & Abelson, 1986; Resnick et al., 2009; Kelleher & Pausch, 2005). And, remarkably, as the 1970s, researchers were thinking about physical computer *languages* that they imagined could open the door to programming for children who were still learning how to read and write (see McNerney, 2004). In all of this work was the democratizing idea that anyone, regardless of age or background, could engage in computational literacy (diSessa, 2000) experiences. Tangibility, whether real or metaphorical, was central to this vision. We'll talk a little about this history in this chapter, but our main focus will be on why tangible computing matters now and how we see it shaping the future landscape of computing education. The chapter will touch on physical computing and robotics, but our main emphasis will be on the use of tangible technologies to support computer programming in and with the physical world. We express cautious optimism about this future. Despite the progress we've made, it's clear that tangible computing as an educational endeavor is still very much in its infancy. This field has potential to invite a diverse new generation into computing and to advance computer science education, but there is still work to be done to fully realize this vision.

2. A Brief History Of Tangible Computing Education

In the years leading up to the turn of the century, the physicality of computing education experienced something of a resurgence through both technological advances as well as the sustained efforts of researchers to advance human interaction with computers beyond the computer screen and into the real world. Ishii and Ullmer (1997) coined the term "tangible" to describe a class of computer interfaces that employ physical objects and surfaces as a means to both manipulate and represent digital information. Their use of the term was meant to capture the idea that much of the richness of human interaction with the physical world through the use of tools has been replaced by uniform interaction with narrow bandwidth input devices such as mice, keyboards, and touchscreens. They were also considering a much older history of computation anchored in physical materials (such as the Abacus). The incorporation of a variety of physical objects and multi-sensory feedback was seen as a way to recapture some of this richness to *humanize* human-computer interaction. Later definitions such as Dourish's notion of *embodied interaction* (Dourish, 2001) and Hornecker and Buur's (2006) *tangible interaction* emphasized the degree to which interactive systems could be meaningfully embedded in physical, social, and cultural contexts. In this sense, tangibility became less about the physical nature of the interface and more about the idea that interaction with digital systems could be entangled within material and social realities beyond that of an individual sitting in front of a computer screen. All of these ideas have roots in the *ubiquitous computing* movement of the late 1980s and early 1990s that imagined a world in which machines would increasingly conform to human dimensions, capabilities, and activity structures rather than the other way around (Weiser, Gold, & Brown, 1999).

Perhaps not surprisingly, much of the research involving tangible interaction has emphasized education and learning (Shaer & Hornecker, 2010; O'Malley & Fraser, 2004; Bers, 2008). For example, the work of Resnick and collaborators at the MIT Media Lab is notable for its focus on *digital manipulatives*, computationally enhanced versions of traditional children's toys that created new opportunities for learners to engage with complex concepts. For example, Digital Beads (Resnick et al., 1998) allowed children to create simple programs in a language of one-dimensional cellular automata by stringing together small capsules with embedded LEDs that could transmit, absorb, or destroy light passed from adjacent beads. The System Blocks project (Zuckerman, Arida, & Resnick, 2005) provided a similar interface for simulating dynamic systems. Wooden blocks with embedded electronics expressed behaviors of complex systems, such as stocks, flows, and feedback loops. The work of this group also helped to open physical computing and robotics to a broader (and younger) audience. Their LEGO/Logo project made it possible for children to write computer programs to control animated LEGO constructions incorporating sensors and motors (Resnick, Ocko, Papert, 1988). This work was followed by other influential projects like LEGO Mindstorms and the MIT Cricket (Resnick et al, 1998; see also Blikstein, 2013). Blikstein's review of physical computing kits (2013) describes four waves of innovation in physical computing education, starting with systems such as LEGO/Logo (Resnick, Ocko, & Papert, 1988) in the 1980s and leading to systems such as the Arduino (Mellis, Banzi, Cuartielles, & Igoe, 2007), PICO Cricket (Rusk et al., 2008), Cubelets (Schweikardt & Gross, 2006), and the LilyPad Arduino (Buechley & Eisenberg, 2008).

In addition to physical computing systems programmed with graphical or text-based languages, researchers have also explored the idea that computer code itself can be represented using physical objects. With a text-based language, programmers use words like BEGIN, IF, and REPEAT to instruct a computer. This code must be written according to strict, and often frustrating, syntactic rules. With a visual or graphical language (see Chapter 3.2 and 3.10), words are replaced by pictures, and programs are expressed by arranging and connecting icons on the computer screen. Syntactic rules can be conveyed to the programmer through a rich set of visual queues based on cultural and diagrammatic conventions. Visual languages can be less intimidating for beginners and have become popular in educational settings (Chapter 3.2). Tangible languages go a step further. Instead of relying on pictures and words on a computer screen, tangible languages use physical objects in the real world to represent various programming elements, data abstractions, and flow-of-control structures. Users manipulate, arrange, and connect these physical elements to construct runnable programs. Rather than relying on implied rules and spatial metaphors, and user interface conventions, tangible languages can exploit the physical properties of objects such as size, shape, and material to express and enforce syntax.

Researchers began exploring the idea of tangible languages as early as the the 1970's. Radia Perlman, then a researcher at the MIT Logo Lab, believed that the syntax rules of text-based computer languages represented a serious barrier to learning for young children. To address this issue she developed an interface called Slot Machines (see McNerney, 2004) that allowed young children to insert cards representing various Logo commands into three colored racks, which in turn represented subroutines.

Almost two decades later projects such as Suzuki and Kato's AlgoBlocks (Suzuki & Kato, 1995) began to revisit these ideas. Since that time, a wide variety of tangible languages have been developed and explored, including projects that blend movement and action and physical space with digital programming (Fernaeus & Tholander, 2006; Sherman et al., 2001), robots that are also embodied algorithmic structures (Schweikardt & Gross, 2006; Wyeth, 2008), the incorporation of found or crafted materials into algorithmic expressions (Smith & Kotzé, 2010), and the integration of physical activity and play with programming (Smith, 2007). Increasingly, tangible languages are making their way out of research labs and into the public sphere in the form of offerings such as museum exhibits, educational tools, and commercial products (Horn, Crouser, & Bers, 2012; Hu, Zekelman, Horn, & Judd, 2015; Sullivan, Bers, & Mihm, 2017)[1]. As these ideas have gained a commercial foothold, research have started to consider the learning affordances of tangible vs screen-based interfaces (Pugnali et al, 2017; Sullivan & Bers, 2017; Strawhacker, & Bers, 2015; Horn et al., 2012; Strawhacker, et al 2013).

In this chapter, we broadly classify tangible languages into three categories: smart block languages, demonstration languages, and externally compiled languages.

[1] See also: https://www.primotoys.com/, https://www.bee-bot.us/bluebot.html, https://www.playosmo.com/en/coding/

Smart Block Languages: Smart block programming languages feature interlocking physical blocks that can be stacked or connected to form a program. In all cases, the blocks themselves contain electronic components or microprocessors, which, when connected, form structures that are more than just abstract representations of algorithms; they form working, specialized computers that can execute code through the sequential interaction of the blocks. Their physical structures embody both the program and the means for its execution. For example, McNerney's Tangible Computation Bricks (2004) embedded Cricket microprocessors into LEGO bricks that could be stacked to form physical algorithmic structures. The bricks also accepted a single parameter card that could interchangeably be a constant, a timer, a sensor, or some user-adjustable value. Along similar lines, Wyeth (2008) created a smart block language for younger children (ages four to eight) also using stackable LEGO-like blocks to describe simple programs. This language consisted of sensor blocks, logic blocks, and action blocks for generating light, sound, and motion. Schweikard and Gross (2006) developed a distributed construction kit system consisting of interlocking cubes that encouraged users to combine sensors, logic elements, and actuators, exposing them to a variety of advanced concepts including kinematics, feedback and distributed control.

Tangible Demonstration Languages: A second class of tangible language allows users to program physical systems or environments by demonstrating a set of rules or actions that can be kinetic, audible, or digital. The computer then repeats these steps to act out a program. Researchers at the University of Maryland have explored approaches for controlling ubiquitous computing environments for storytelling (Sherman et al, 2001). Working with children, the researchers developed and evaluated demonstration-based programming systems called StoryKits. Using a "magic wand," young children (ages four to six) were able to program the various props and physical icons that made up their story worlds. In a slightly different direction, Frei, Su, Mikhak, and Ishii designed an educational toy called Curlybot (Frei, Su, Mikhak, & Ishii, 2000) that could record and play back its motion on a flat surface. Children could program the robot by dragging the robot to demonstrate motion. Taking this idea of kinetic memory further, Raffle, Parkes, and Ishii created Topobo (2004), a system that allows children to construct imaginative creatures composed of passive and active building components. The active components have the ability to record and playback physical motion, helping children learn about animal movement and their own bodies in the process.

Externally Compiled Languages: Unlike smart block languages, programs created with a externally compiled tangible language are only symbolic representations of actual algorithms—much in the way that Java or C++ programs are only collections of text files. An additional piece of technology (a compiler or interpreter), must be used to translate the abstract representations of the program into a machine language that will be executed on some computer system. Ideally, the tangible elements of a compiled language contain little or no electronic components, affording the language designers more freedom in the choice of objects and materials to work with. For instance, paper or flat cards with attached RFID tags become realistic options. Early examples of such languages come from Horn and colleagues with projects like Quetzal and Tern (Horn & Jacob, 2007). These languages use passive tangible objects encoded with computer vision fiducials that allow a camera to translate blocks into working programs. Researchers at the DevTech research group at Tufts University, developed KIBO (Bers, 2018a; Sullivan, Bers, & Mihm, 2017), a robot that can be programmed with wooden blocks with barcodes. Children assemble the robot by incorporating its sensors, motors and art platforms, and then utilize the embedded barcode scanner to compile their programs, block by block, creating a direct link between the robot and the program it executes. Newer systems have made use of augmented reality and video-based object tracking (Hu et al., 2015) to combine some of the real-time interaction of smart block languages with the practical advantages of passive tangible objects.

Each of the three classifications of tangible languages: smart block languages, demonstration languages, and externally compiled languages, present their own challenges and affordances and can better serve different populations of learners with their own unique needs and developmental capabilities.

3. Why Tangibility Matters (Four Themes)

Given the history and accelerating interest in tangible and physical computing for education, a reasonable question to ask is why it all matters? Creating tangible materials comes with associated costs that simply aren't a factor for pure software systems. While software can be deployed online, physical materials have to be designed, manufactured, and distributed. These costs will decrease given advances in 3D printing, homemade electronics, and the increasing availability of makerspaces, but it is still worth asking what we gain from tangibility that makes added costs worthwhile? A tempting answer is that physicality confers a certain degree of cognitive leverage in learning situations, especially for younger children. Variants of this argument, usually anchored in notions of sensorimotor engagement with the material world, have been around for decades[2]. For example, research based on *conceptual metaphor theory* (Lakoff & Johnson, 2008) has argued that learning experiences that make use of physical properties of materials, movement through space, and relationships between objects and people might more successfully reference sensorimotor schema that form the foundation for much of abstract thought (e.g. Hurtienne & Israel, 2007; Macaranas et al., 2012). While we agree that it is appealing to consider cognitive benefits of tangibles for learning, here we propose four other broad themes that have perhaps received less attention in the literature but

[2] Literature on manipulative materials in early mathematics education is an interesting case that has had mixed results (see Uttal, D. H., Scudder, K. V., & DeLoache, J. S., 1997)

nonetheless illustrate what we see as important future directions of tangibility in CS Education research. These themes have to do with broadening the access to and appeal of computational literacy experiences, in part by making them more universal and visible. For each theme, we highlight example projects that illustrate the potential of tangibles in the future of CS education.

3.1 Theme 1: Early Childhood Learning

The world of early childhood education often privileges children's engagement in rich sensorimotor experiences with both the natural world and physical materials, while remaining cautious about "exposure" to digital media and screen time (American Academy of Pediatrics, 2016). In this context, the development of programming languages that make it possible for children to code with objects such as wooden blocks, tiles, beads, or even craft materials, has helped reimagine computational thinking as a developmentally appropriate activity that can be integrated with other classroom experiences in a natural way (Bers, 2008; Bers, 2018a; Bers & Horn, 2009). Using programming and robotics systems, children can program, debug, and play with concepts like sequences, patterns, logic, loops, sensors, and actuators, often without ever interacting with screen-based media (Figure 22.1 and 22.2).

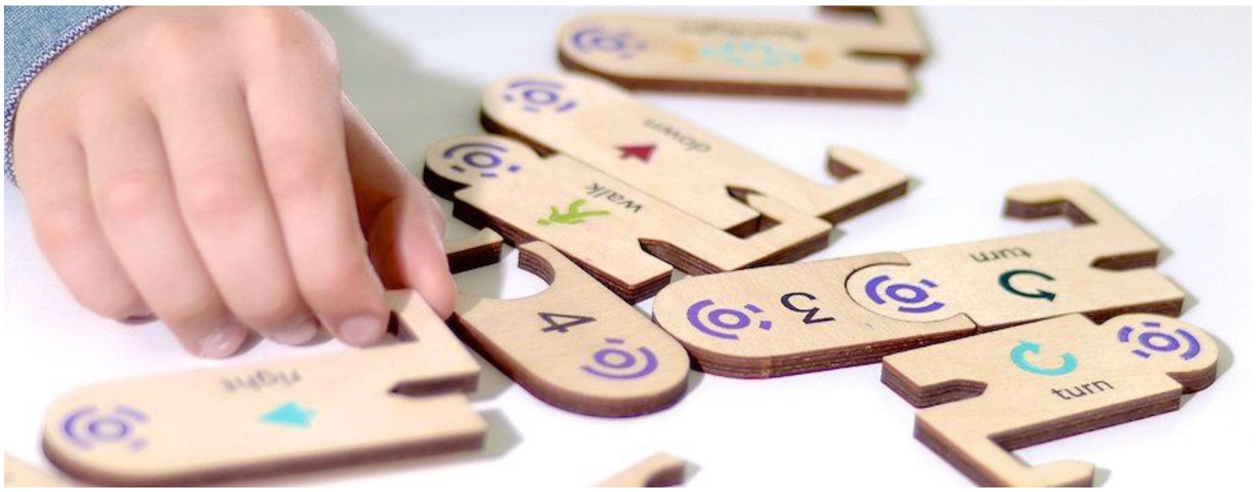

Figure 22.1 A prototype tangible programming language based on computer vision technology. Image credit Felix Hu.

One of the earliest examples is KIBO, a robotics and tangible programming kit for children ages 4-7 years old. KIBO started as a research project at the DevTech research group at Tufts University and became a commercially available product in 2014 (Bers, 2018a). KIBO lets children build their own robots, decorate them with art supplies, and program them, without requiring PCs, tablets, or smartphones. To program their robotic creations, children put together sequences of instructions (programs) using the wooden KIBO blocks that they can then scan with a barcode reader built into the body of the robot (see Figure 22.2). The language syntax in KIBO (i.e. a sequential connection of blocks) is designed to support and reinforce sequencing skills in young children (Bers, 2018a; Horn, Crouser, & Bers, 2012).

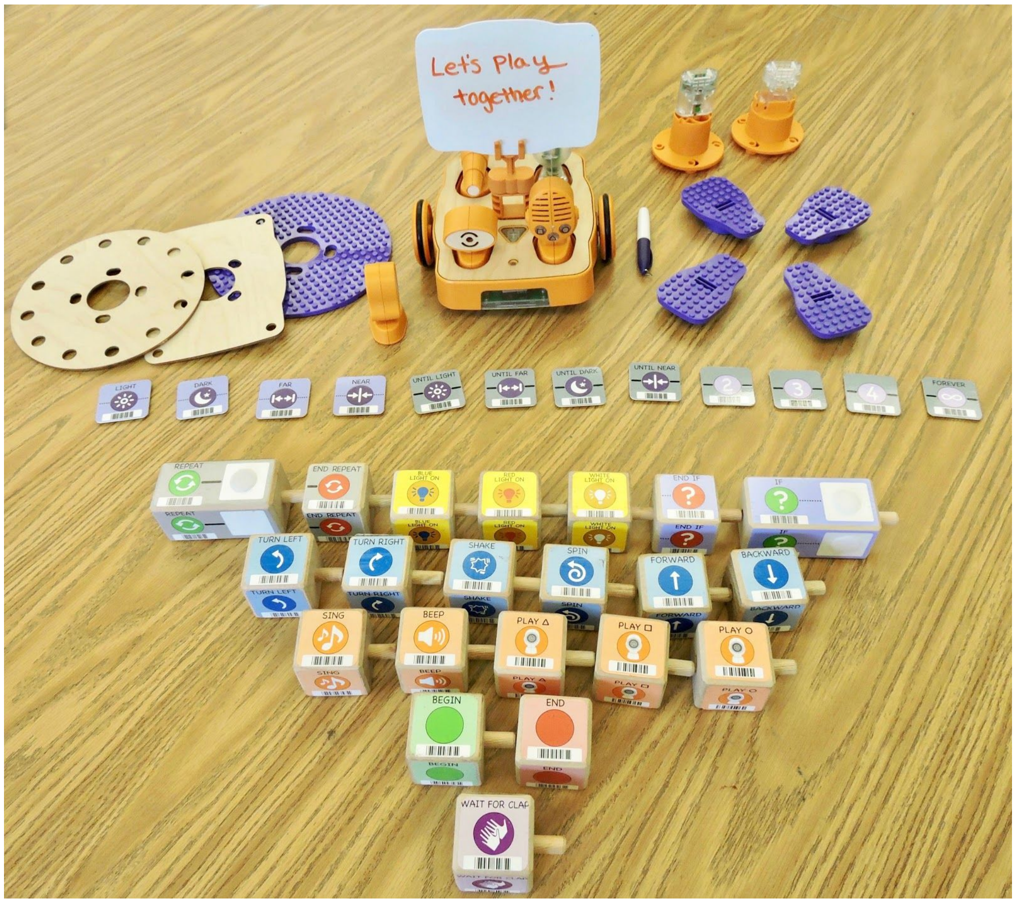

Figure 22.2 KIBO robot and its blocks. Reproduced with permission from Bers (2018b).

Research with KIBO in early childhood classrooms has shown that children, as young as pre-school, were able to create sequential programs as well as more sophisticated algorithms that utilize control structures with number and sensor parameters (Sullivan & Bers, 2015; Sullivan & Bers, 2017). Research also shows statistically significant improvement of kindergarten children in sequencing skills, which are predictors of later numeracy and literacy (Kazakoff & Bers2012).

Beyond classrooms, researchers are using similar approaches to introduce foundational computational literacy experiences for young children through culturally relevant artifacts. For example, Horn et al. (2013) explored the use of coding “stickers” embedded in a children’s storybook as a way to engage parents and children together in playful computer programming activities. The technology combined a paper storybook with computer programming activities that children complete by adhering stickers to the pages of the book. The programs then controlled an interactive digital character that appeared on the screen of a smartphone or tablet computer. The researchers argued that children’s storybooks are a powerful *cultural form* of literacy that support subtle but powerful parent-child reading practices that scaffold parental involvement in children’s early *computational* literacy activities.

The designers of these and similar systems are intentionally shaping materials to better speak the language of early childhood. In this context, educators can start to blend computational artifacts within a broader child-driven inquiry model. Tangible materials and technologies will continue to create opportunities to support learning with younger children, enabling designers to think about what emerging computational literacy might look like. However, the accelerated availability of commercial products claiming that they can engage young children in learning about computer science, while most of them only provide a limited “playpen” as opposed to an open-ended “playground” (Bers, 2018a; Bers, 2012) might invite researchers and policy makers to examine what are the minimal design features of such environments to claim that they support learning the sequential, algorithmic, and problem skills associated with programming, as well as afford expressiveness of ideas through projects that are personally meaningful to children.

3.2 Theme 2: Appealing to a Broader Audience

The inability of CS education to attract and retain diverse learner participation, both in schools and beyond, is a persistent and discouraging trend that has been taken up multiple times in this book already (Chapters 3.5, 3.13). It is possible that we are now at a turning point with new and more inclusive learning environments and programs that are connecting with learners from backgrounds previously underrepresented in computer science and related fields. However, the weight of evidence from post-secondary degree programs suggests that we still have much work to do (Zweben & Bizot, 2016). One of the most appealing aspects of tangible computing is its potential to connect with a broader audience, representing a more diverse range of cultural traditions, practices, and value systems.

In the realm of physical computing, projects like LilyPad Arduino (Buechley & Eisenberg, 2008; Searle et al., 2014; Kafai et al., 2014) and e-textiles are using innovative designs to integrate computational and electronic materials with craft traditions such as sewing, scrapbooking, drawing, music, and fashion. Such craft traditions have rich cultural roots characterized by multigenerational communities of practice. The transformation here, although perhaps not fully realized, is subtle. It’s not that computation is being superficially dressed up in new clothes so as to become more palatable to a diverse audience; rather, these toolkits and materials are ideally appropriated by existing communities as a new medium of expression, grounded in evolving communities and practices. These new systems help illustrate the potential relevance of CS Education in a much broader array of activities and endeavors (including craft traditions, music, dance, fashion, visual arts, storytelling, and so on).

Another example of this potential comes from the work of Horn and colleagues on a tangible programming and robotics exhibit that was installed at the Museum of Science, Boston. The exhibit allowed visitors to control the movement of a robot on a platform by constructing programs from chains of wooden blocks shaped like jigsaw puzzle pieces. Compared to a version of the exhibit in which visitors used a computer mouse to program in the same robot, children (and girls in particular) were significantly more likely to try the exhibit in the tangible condition (Horn et al., 2012). And, although visitors created similar programs in the two conditions, they were also more likely to engage in collaborative exploration with other family members in the tangible condition.

Tangible approaches have also shown potential for children with a range of cognitive and physical abilities. KIBO was used with children with Autism spectrum disorder and preliminary results from a pilot study in Panama show that children were not only able to successfully program their robots and understand their code in order to debug it, but also engage in social interactions through the activity of arranging blocks in a sequence (Albo-Canals, et al, in press). Other researchers have explored affordances of tangible programming languages for children with visual impairments (Thieme et al., 2017).

3.3 Theme 3: Increasing the Visibility of Digital Artifacts in Learning Spaces

Robotics activities in educational settings are characterized by the creative chaos of building, testing, failure, rebuilding, and refining. Children move back and forth between programming stations (such as laptop computers) where they build and refine simple computer programs that provide the logical glue between sensors (touch, light, etc.) and actuators (motors, sounds, lights). One wonderful aspect of this kind of work is that robots and the construction process have a physical presence that is highly visible. But this visibility also highlights the relatively little attention that software (the programs that students construct to control the robots) receives compared to hardware. The robots are colorful, physical creatures that come alive with light, sounds, and motion. Computer code, on the other hand, often lives a transient and anonymous life on a computer display. It’s much harder to see, it gets covered up by other windows of digital content, and it is often forgotten altogether. After the lights turn off and the children go home, the robots are often still displayed in the classroom, while the code that made the robots run is nowhere to be seen. This can happen across a variety of CS education activities, including animation, graphics, video games, e-textiles, and robotics competitions. In the landscape of computational artifacts, code can become a forgotten, second-class citizen.

One appealing aspect of tangible programming languages is that they give “code” a persistent, physical life in the learning space. Code has the potential to become a structure that can be literally held on to as a robot executes its program, and something that is persistently visible and available for the same level of tinkering, refinement, and debugging as the robot itself. This tangibility also facilitates debugging in a social context. Children can see what doesn’t work, and they can help fix it. Depending on the system, the programs might take a variety of forms: stickers on a piece of paper, interlocking wooden blocks, magnetic tiles on a whiteboard, or even Lego bricks that piggyback on the robot itself. The added visibility of code might bring attention to previously neglected properties and concepts such as elegance, the importance of debugging, and testing for edge cases and other unusual situations. In other words, code has the potential to become an object of conversation and attention in a way that it might not have been before.

3.4 Theme 4: Beyond toys and games

Although we see much promise in tangible computing, most of the existing work focuses on younger children, with relatively less attention directed towards older learners. The reasons are varied. The push away from predominantly screen-based media is a move that resonates with early childhood education, and for young children with developing literacy and fine motor skills, physical materials offer an appealing and perhaps more accessible entry point into computational literacy. However, another reason has to do with current technical limitations of tangible interfaces and their fit for more advanced educational, professional, and real-world settings. While touch-sensitive devices and computer displays have made tremendous strides in the past 20 years, tangible interfaces still predominantly rely on computer vision techniques or embedded electronic components. Computer vision suffers from usability problems when a camera can't clearly see target objects. On the other hand, objects with embedded electronics are relatively difficult and expensive to manufacture and depend on pieces having power and electrical connections. This contrasts with the needs of even moderately sophisticated visual languages that provide learners with dozens of distinct programming elements that must be organized through a menu system. Saving and restoring programs is also difficult, at least with today's technology because physical structures are hard to automatically rebuild. For the same reasons, the clipboard functions (e.g. copy, cut, paste, and delete) of text-based and visual programming languages would be difficult to implement with most tangible languages, making programming tedious for more advanced programmers who want to quickly copy and modify existing pieces of code.

Along with these barriers is the perception that *real* coding is done in text-based languages, preferably on editors with dark background and candy-colored syntax highlighting. This perception has shifted as visual languages have grown in popularity and now appear in high school and even college-level curricula. But, there is still a dominant view that the authentic programming of engineers and other practitioners is done in text.

And yet, even with this laundry list of limitations, technology will continue to advance, making the tedious, flaky, or impossible of today much more appealing, practical, and reliable tomorrow. As an obvious example, dramatic improvements in augmented reality will open many new possibilities for learners to collaboratively build and debug programs created with low-cost and generic physical objects. These advancements will expand the role for tangible programming languages in the areas of education and non-professional programming situations, and not just for younger children, or even children at all.

One obvious area for innovation is in end-user programming systems. In the age of smart technology (smart homes, smart thermostats, smart cars, the Internet of Things, and ecosystems of connected devices), the opportunities and need for end-user-programming environments will proliferate (see Myers, Ko, & Burnett, 2006; Blackwell & Hague, 2001). While the control of an individual device may be limited to simple configurations of input options and settings ("when it's after 8pm on a Wednesday, turn the heat down by 10 degrees"), in an increasingly ubiquitous world, the complexity of simple interconnected devices will multiply. This complexity will manifest itself in multiple spheres, including (and perhaps especially) in social spheres that play out between family members, coworkers, neighbors, and citizens. The ability for end users to write programs, even programs consisting of simple interdependent conditions and outcomes, might be necessary to manage the myriad unanticipated situations that arise when such systems are deployed on a large scale. As we're seeing in the current debate around the role of algorithms in society (see Mittelstadt et al., 2016), we need to think carefully about issues of power and accessibility when it comes to end-user programming systems. While the algorithms that control social media news feeds impact millions of users, end user programming might intersect with the social dynamics of a family, school, or workspace. This "programming" might look quite different from languages used in introductory computer science classes, but there are also certainly rich opportunities for computational thinking. For example, imagine a family argument over conflicting rules given to a programmable thermostat. How are these rules resolved by the system, and what rules *should* take precedent to make family members happy and comfortable? More to the point, who has the power to make decisions on behalf of the family and how visible are these decisions? Family members might have different opinions about temperature and time settings (when is bedtime, and how much should we adjust the temperature?). But, going beyond the case of household heating and cooling systems, there are many examples of potential end-user programming systems that we could imagine becoming tangible. Beyond smart homes, there are many other domains where tangible programming might make sense in professional and creative settings. For example, musicians, DJs, and visual performance artists often write live code (see Blackwell, McLean, Noble, & Rohrhuber, 2014 for an extensive overview) that becomes an integral part of the performance itself. We've already seen tangible systems in this space (e.g. Xambó, 2017) and the area seems ripe for exploration.

4. Conclusion

Advancements in technology will continue to drive new forms of human interaction with computational systems. With these changes will come new opportunities for Computer Science education to reach broader audiences and engage learners in new ways. In the past several years we have already seen the outpouring of new tangible programming products marketed to young children and their parents. There are a wealth of research opportunities in this space as well as a need for thoughtful reflection and study on understanding both the benefits and limitations of tangible technologies. In this chapter we have reviewed four potential themes that suggest the relevance of tangible computing for the CS Education Research community: thinking about CS education in early childhood, reaching and engaging more diverse learners, increasing the visibility of computer code in educational practice, and engaging broader audiences beyond traditional education settings. This is not an exhaustive list, but it hints at the broad array of research questions yet to be addressed.

Acknowledgements

Much of this work was made possible through funding from the National Science Foundation (grants DRL-1451762, IIS-0414389, DRL-0735657, DRL-1118897, DRL-0735657). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Albo-Canals, J., Barco, A., Relkin, Hannon, D., Heerink, M., Heinemann, M., Leidl, K. & Bers, M. (In press) The Use Case of KIBO Robot to Positively Impact Social and Emotional Development in Children with ASD; *Journal of Social Robots.*

American Academy of Pediatrics. (2016). Media and young minds. *Pediatrics*, **138**(5), e20162591.

Bers, M (2008). *Blocks to Robots: Learning with Technology in the Early Childhood Classroom*, Teachers College Press.

Bers, M. U. (2012). *Designing digital experiences for positive youth development: From playpen to playground*, Cary, NC: Oxford.

Bers, M.U., (2018a). C*oding as a Playground: Programming and Computational Thinking in the Early Childhood Classroom*, New York, NY: Routledge press.

Bers, M. U. (2018b). Coding, playgrounds and literacy in early childhood education: The development of KIBO robotics and ScratchJr. In *Global Engineering Education Conference (EDUCON)* (pp. 2094-2102). New York, NY: IEEE.

Bers, M. U., & Horn, M. S. (2010). Tangible programming in early childhood, In Berson, I.R. and Berson, M.J. eds., *HighTech Tots: Childhood in a Digital World*. Charlotte, NC: IAP, pp. 49-70.

Blackwell, A.F. and Hague, R., 2001. AutoHAN: An architecture for programming the home. In *Human-centric computing languages and environments, 2001. Proceedings IEEE symposia on* (pp. 150-157). New York, NY: IEEE.

Blackwell, A., McLean, A., Noble, J., & Rohrhuber, J. (2014). Collaboration and learning through live coding (Dagstuhl Seminar 13382). In *Dagstuhl Reports* (Vol. 3, No. 9). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Blikstein, P. (2013). Gears of our childhood: constructionist toolkits, robotics, and physical computing, past and future. In *Proceedings of Interaction Design and Children* (pp. 173-182). New York, NY: ACM Press.

Buechley, L., & Eisenberg, M. (2008). The LilyPad Arduino: Toward wearable engineering for everyone. *IEEE Pervasive Computing,* **7**(2), 12-15.

diSessa, A. A. (2001). *Changing minds: Computers, learning, and literacy*. Cambridge, MA: MIT Press.

diSessa, A. A., & Abelson, H. (1986). Boxer: A reconstructible computational medium. *Communications of the ACM,* **29**(9), 859-868.

Dourish, P. (2004). *Where the action is: the foundations of embodied interaction*. Cambridge, MA: MIT press.

Erwin, B., Cyr, M., & Rogers, C. (2000). Lego engineer and robolab: Teaching engineering with labview from kindergarten to graduate school. *International Journal of Engineering Education,* **16**(3), 181-192.

Fernaeus, Y., & Tholander, J. (2006). Finding design qualities in a tangible programming space. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 447-456). New York, NY: ACM Press.

Frei, P., Su, V., Mikhak, B., & Ishii, H. (2000). Curlybot: designing a new class of computational toys. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 129-136). New York, NY: ACM Press.

Horn, M. S., AlSulaiman, S., & Koh, J. (2013). Translating Roberto to Omar: computational literacy, stickerbooks, and cultural forms. In *Proceedings of Interaction Design and Children* (pp. 120-127). New York, NY: ACM Press.

Horn, M. S., Crouser, R. J., & Bers, M. U. (2012). Tangible interaction and learning: the case for a hybrid approach. *Personal and Ubiquitous Computing,* **16**(4), 379-389.

Horn, M. S., & Jacob, R. J. (2007). Designing tangible programming languages for classroom use. In *Proceedings of Tangible and Embedded Interaction* (pp. 159-162). New York, NY: ACM Press.

Hornecker, E., & Buur, J. (2006). Getting a grip on tangible interaction: a framework on physical space and social interaction. In *Proceedings of the SIGCHI conference on Human Factors in computing systems* (pp. 437-446). New York, NY: ACM Press.

Hu, F., Zekelman, A., Horn, M., & Judd, F. (2015). Strawbies: explorations in tangible programming. In Proceedings of *Interaction Design and Children* (pp. 410-413). New York, NY: ACM Press.

Hurtienne, J., & Israel, J. H. (2007). Image schemas and their metaphorical extensions: intuitive patterns for tangible interaction. In *1st international conference on Tangible and embedded interaction* (pp. 127-134). New York, NY: ACM.

Ishii, H., & Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI Conference on Human factors in computing systems* (pp. 234-241). New York, NY: ACM Press.

Kafai, Y., Searle, K., Martinez, C. and Brayboy, B. (2014). Ethnocomputing with electronic textiles: culturally responsive open design to broaden participation in computing in American indian youth and communities. In *Proceedings of the ACM Technical Symposium on Computer Science Education* (pp. 241-246). New York, NY: ACM Press.

Kazakoff, E., & Bers, M. (2012). Programming in a robotics context in the kindergarten classroom: The impact on sequencing skills. *Journal of Educational Multimedia and Hypermedia,* **21**(4), 371-391.

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR),* **37**(2), 83-137.

Lakoff, G., & Johnson, M. (2008). *Metaphors we live by*. University of Chicago Press.

Macaranas, A., Antle, A. N., & Riecke, B. E. (2012). Bridging the gap: attribute and spatial metaphors for tangible interface design. In *Sixth International Conference on Tangible, Embedded and Embodied Interaction* (pp. 161-168). New York, NY: ACM Press.

McNerney, T. S. (2004). From turtles to Tangible Programming Bricks: explorations in physical language design. *Personal and Ubiquitous Computing,* **8**(5), 326-337.

Mellis, D., Banzi, M., Cuartielles, D., & Igoe, T. (2007). Arduino: An open electronic prototyping platform. In *Proceedings SIGCHI Conference on Human Factors in Computing Systems (extended abstracts)*. New York, NY: ACM Press.

Mittelstadt, B. D., Allo, P., Taddeo, M., Wachter, S., & Floridi, L. (2016). The ethics of algorithms: Mapping the debate. *Big Data & Society*, **3**(2), 1-21.

Myers, B. A., Ko, A. J., & Burnett, M. M. (2006). Invited research overview: end-user programming. In *Proceedings SIGCHI Conference on Human Factors in Computing Systems (extended abstracts)*. New York, NY: ACM Press.

Claire O'Malley, Danae Stanton Fraser. (2004). *Literature Review in Learning with Tangible Technologies. A NESTA Futurelab Research report - report 12. 2004*

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas.* New York: Basic Books.

Pugnali, A., Sullivan, A., & Bers, M.U. (2017) The Impact of User Interface on Young Children's Computational Thinking. J*ournal of Information Technology Education: Innovations in Practice,* **16**, 172-193.

Pattis, R. E. (1981). *Karel the robot: a gentle introduction to the art of programming,* Hoboken, New Jersey: John Wiley & Sons, Inc.

Raffle, H. S., Parkes, A. J., & Ishii, H. (2004). Topobo: a constructive assembly system with kinetic memory. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 647-654). New York, NY: ACM Press.

Repenning, A. (1993). Agentsheets: a tool for building domain-oriented visual programming environments. In *Proceedings of the INTERACT'93 and CHI'93 Conference on Human Factors in Computing Systems* (pp. 142-143). New York, NY: ACM Press.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K. & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM,* **52**(11), 60-67.

Resnick, M., Martin, F., Berg, R., Borovoy, R., Colella, V., Kramer, K., & Silverman, B. (1998). Digital manipulatives: new toys to think with. In Proceedings of the S*IGCHI Conference on Human Factors in Computing Systems* (pp. 281-287). New York, NY: ACM Press.

Resnick, M., Ocko, S., & Papert, S. (1988). LEGO, Logo, and design. *Children's Environments Quarterly*, **5**(4), 14-18.

Rusk, N., Resnick, M., Berg, R., & Pezalla-Granlund, M. (2008). New pathways into robotics: Strategies for broadening participation. *Journal of Science Education and Technology*, **17**(1), 59-69.

Schweikardt, E., & Gross, M. D. (2006). roBlocks: a robotic construction kit for mathematics and science education. In *Proceedings of Multimodal Interfaces* (pp. 72-75). New York, NY: ACM Press.

Searle, K.A., Fields, D.A., Lui, D.A., and Kafai, Y.B. (2014). Diversifying high school students' views about computing with electronic textiles. In *Proceedings of International Computing Education Research* (pp. 75-82). New York, NY: ACM Press.

Shaer, O., & Hornecker, E. (2010). Tangible user interfaces: past, present, and future directions. *Foundations and Trends in Human-Computer Interaction,* **3**(1–2), 1-137.

Sherman, L., Druin, A., Montemayor, J., Farber, A., Platner, M., Simms, S., ... & Kruskal, A. (2001). StoryKit: tools for children to build room-sized interactive experiences. In *SIGCHI Conferene on Human Factors in Computing Systems* (pp. 197-198). New York, NY: ACM Press.

Smith, A. C. (2007). Using magnets in physical blocks that behave as programming objects. In *Proceedings of the 1st international conference on Tangible and embedded interaction* (pp. 147-150). New York, NY: ACM Press.

Smith, A. C., & Kotzé, P. (2010). Indigenous African artefacts: Can they serve as tangible programming objects? In *IST-Africa, 2010* (pp. 1-11). New York, NY: IEEE.

Strawhacker, A. L., & Bers, M. U. (2015). "I want my robot to look for food": Comparing children's programming comprehension using tangible, graphical, and hybrid user interfaces. *International Journal of Technology and Design Education,* **25**(3), 293-319.

Strawhacker, A., Sullivan, A., & Bers, M. U. (2013). TUI, GUI, HUI: Is a bimodal interface truly worth the sum of its parts? In *Proceedings of Interaction Design and Children* (pp. 309-312). New York, NY: ACM Press.

Sullivan, A. and Bers, M.U. (2016). Robotics in the early childhood classroom: learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education*, **26**(1), 3-20.

Sullivan, A. and Bers, M.U. (2017). Dancing robots: integrating art, music, and robotics in Singapore's early childhood centers. *International Journal of Technology and Design Education*, 1-22. DOI 10.1007/s10798-017-9397-0

Sullivan, A. A., Bers, M. U., & Mihm, C. (2017). Imagining, Playing, and Coding with KIBO: Using Robotics to Foster Computational Thinking in Young Children. In *Proceedings of the International Conference on Computational Thinking Education* (pp. 110-115). The Education University of Hong Kong, Hong Kong.

Sullivan, A. & Bers, M.U. (2017). Computational Thinking and Young Children: Understanding the Potential of Tangible and Graphical Interfaces. In H. Ozcinar, G. Wong, & T. Ozturk, eds., *Teaching Computational Thinking in Primary Education*. Hershey, PA: IGI Global, pp. 123-137.

Suzuki, H., & Kato, H. (1995). Interaction-level support for collaborative learning: AlgoBlock—an open programming language. In *The First International Conference on Computer Support for Collaborative Learning* (pp. 349-355). Hillsdale, NJ: Lawrence Erlbaum Associates.

Thieme, A., Morrison, C., Villar, N., Grayson, M., & Lindley, S. (2017). Enabling collaboration in learning computer programing inclusive of children with vision impairments. In *Proceedings of the 2017 Conference on Designing Interactive Systems* (pp. 739-752). New York, NY: ACM Press.

Uttal, D. H., Scudder, K. V., & DeLoache, J. S. (1997). Manipulatives as symbols: A new perspective on the use of concrete objects to teach mathematics. *Journal of applied developmental psychology,* **18**(1), 37-54.

Weiser, M., Gold, R., & Brown, J. S. (1999). The origins of ubiquitous computing research at PARC in the late 1980s. *IBM Systems Journal*, **38**(4), 693-696.

Wyeth, P. (2008). How young children learn to program with sensor, action, and logic blocks. *Journal of the Learning Sciences,* **17**(4), 517-550.

Xambó, A., Drozda, B., Weisling, A., Magerko, B., Huet, M., Gasque, T., & Freeman, J. (2017). Experience and Ownership with a Tangible Computational Music Installation for Informal Learning. In *Tangible and Embedded Interaction* (pp. 351-360), New York, NY: ACM Press.

Zuckerman, O., Arida, S., & Resnick, M. (2005). Extending tangible interfaces for education: digital montessori-inspired manipulatives. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 859-868). New York, NY: ACM Press.

Zweben, S. & Bizot, B. (2016). Taulbee Survey. *Computing Research News,* **29**(5), 3-51.