2 PLAYGROUNDS AND MICROWORLDS: LEARNING TO CODE IN EARLY CHILDHOOD

Marina Umaschi Bers

PROGRAMMING THE "HOKEY POKEY"

In my book *Coding as a Playground* (Bers, 2018), I told the story of Maya and Natan, who are in kindergarten. They are working on a joint KIBO robotics project. KIBO is a robot kit developed in my DevTech research group at Tufts University. It belongs to the family of robotic-based constructionist programming environments, inspired by the Logo turtle and the LEGO Mindstorms robotic kit, but it is explicitly designed for children four to seven years old. KIBO has a programming language made of wooden blocks with pegs and holes that can be inserted into each other to form a tangible sequence of commands. No screens, tables, or keyboards are needed. Each block has an icon, as well as text, representing an instruction: forward, shake, wait for clap, light on, beep, and so on. In addition, each block has a unique bar code. These bar codes are read by an embedded scanner in the KIBO robot (see figure 2.1). That is how coding happens.

During class, Maya and Natan's teacher invited students to program their KIBOs to dance the "Hokey Pokey." Maya carefully chooses the blocks to use. She starts by picking the green "begin" block and concludes with the red "end" block but needs other blocks in between. Those are the ones that will tell KIBO how to dance. She forgot the KIBO "Hokey Pokey" song the teacher taught them, so she is at a loss about what blocks to choose. Natan, her teammate, reminds her of the song:

> You put your robot in
> You put your robot out
> You put your robot in
> And you shake it all about
> You do the Hokey Pokey
> and you turn yourself around
> That's what it's all about!

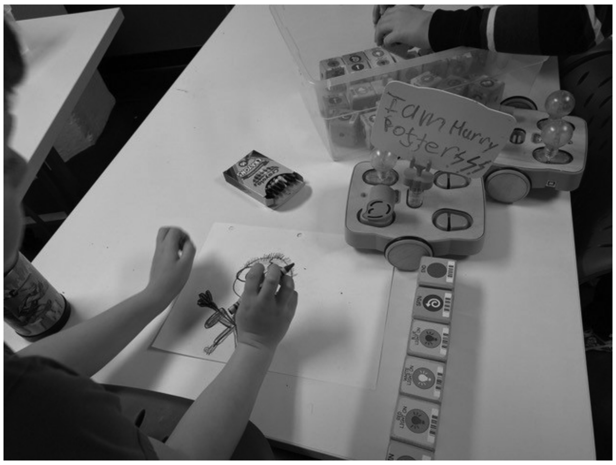

FIGURE 2.1
The KIBO robot.

Maya sings along and, as the song progresses, she chooses the blocks and starts putting them together in a sequence. Begin, "You put your robot in" forward, "You put your robot out" backward, "You put your robot in" forward, "And you shake it all about." She suddenly stops and says, "Natan, I can't find the 'do the Hokey Pokey' block!" "There is no block for that, silly," responds Natan. "We need to make it up. Let's have KIBO turn on the blue and red light instead. That will be our 'Hokey Pokey' block." Maya agrees, adds those two blocks to the sequence, and also adds "shake," "spin," and "beep" to represent the "what it's all about" part of the song. Maya and Natan look at their program while singing the song to make sure they have put together all the needed blocks, in the right order. Then they turn on KIBO for testing. The red light of KIBO's scanner (the "mouth," as Maya calls it) is flashing, meaning that the robot is ready to scan each of the bar codes printed on the wooden programming blocks.

Natan takes his turn and scans the blocks one by one. He goes too fast and skips the "red light" block. Maya points that out and he restarts the scanning. The children are excited to see their robot dance the "Hokey Pokey." "When I count to three, you start singing," says Maya to Natan. They know the drill. They have practiced it during technology circle time in class. Natan sings and both KIBO and Maya dance the "Hokey Pokey." It goes too fast. KIBO dances too fast. "Can you sing faster?" asks Maya. Natan tries one more time, but it still doesn't work. "We have a problem," he says. "I can't sing fast enough to keep up with KIBO." Maya has an idea. For each action in the song, she puts two blocks so that KIBO's motions will last longer. For example, for the "you put your robot in" part, instead of just one "forward" block, she puts two forward, and so on, for each of the commands. Natan tries singing again, and this time KIBO dances at the right pace.

Both children start clapping, shaking their bodies, and jumping up and down. Without knowing it, they engaged with many powerful ideas of computer science, such as sequencing, algorithmic thinking, and debugging or problem solving. They also explored math concepts they are learning in kindergarten, such as estimation, prediction, and counting. Furthermore, they engaged in collaboration. "Children already have too much screen time at home. When they are at school, I want them learning new concepts and skills in STEM, but as importantly, I want them learning to socialize and collaborate with others. I want them looking at each other, and not at the screen," explains Marisa, their kindergarten teacher. "KIBO is just perfect for that." During technology circle time, Marisa asks every group of children to give a demo of their dancing KIBOs. Everyone else is invited to stand up and dance alongside. There is laughter and clapping. There is physical activity, socialization and language development, problem solving, and creative play. It is fun. It feels like a playground, not a coding class.

I coined the metaphor "playground vs. playpen" (Bers, 2012) to discuss the role that new technologies can have in young children's lives. Playgrounds are open ended. Playpens are limited. Playgrounds invite fantasy play, imagination and creativity, social interaction, and teamwork; they require conflict resolution and little adult supervision. The story of Maya and Natan provides an illustration of what learning looks like within a coding playground. In my work at Tufts University, my colleagues and I focus on designing experiences and programming environments that support a playground approach. Our work with KIBO and ScratchJr are some examples (Bers, 2018). In this chapter, I further examine the concept of coding playgrounds and explore the similarities and differences between coding playgrounds and the concept of microworlds, as described by Papert.

CODING PLAYGROUNDS AND MICROWORLDS

The "playground vs. playpen" metaphor provides a way to understand the kind of developmentally appropriate experiences that new technologies, such as programming languages, can promote: problem solving, imagination, cognitive challenges, social interactions, motor skills development, emotional exploration, and making different choices. In order to understand playgrounds, and their relationship with microworlds, we first need to understand playpens. In contrast to the open-ended playground, playpens convey lack of freedom to experiment, lack of autonomy for exploration, lack of creative opportunities, and lack of taking risks. Although playpens are safer, playgrounds provide infinite possibilities for growth and learning. Microworlds are closer to playgrounds than to playpens. However, there are some differences.

Papert (1980) described microworlds as:

> A subset of reality or a constructed reality whose structures matches that of a given cognitive mechanism so as to provide an environment where the latter can operate effectively. The concept leads to the project of inventing microworlds so structured as to allow a human learner to exercise particular powerful ideas or intellectual skills (p. 204).

In this definition, a microworld, like a playground, presents a subset of a reality, a subset so carefully chosen that its structures are explicitly designed to encourage children to engage with a particular set of powerful ideas. Programming languages, such as KIBO, ScratchJr, or Logo, are microworlds with structures called programming blocks, scripts or commands. When these are put together following an orderly sequence (an algorithm), we can observe a range of possible behaviors. While coding, children exercise particular powerful ideas or intellectual skills. Those are associated with what researchers call now computational thinking (Wing, 2006). The cognitive mechanisms that Papert refers to are sequencing, abstraction, modularization, problem solving, and logical thinking.

A playground can also be seen as a microworld. As such, it is a subset of reality that presents itself with structures carefully chosen to encourage children to encounter a particular set of powerful ideas. For example, climbing structures, slides, seesaws, and swings allows children to explore ideas from physics. However, the powerful ideas children encounter when visiting the playground go beyond the cognitive domain. At the playground, children also encounter ideas relevant at the personal, social, emotional, and moral domains. Coding playgrounds, in contrast with Papert's microworlds, reinforce the notion of a "whole child," not only a thinking child. This child learns about the social world by negotiating for her favorite toys in the sandbox, about her own emotions when she struggles to keep up with others on the monkey bars, about moral choices and consequences when she is faced with the dilemma to wait politely for her turn on the swing or cut the existent line. In the playground, this child is encountering all of the dimensions of human development. However, she is doing it in a safe space, a place where she can make mistakes and learn how to try again.

She has autonomy to discover her own way of doing things and to ask for help when needed. Usually the adults are nearby, sitting on a bench and talking with each other.

When Papert developed the concept of microworlds, he was heavily influenced by his experience working with Piaget. Thus, the emphasis was on the cognitive dimension and the methodological approach of exploratory, hands-on learning. Within this perspective, programming languages such as Logo provided an innovative opportunity to create microworlds to support active learning by creating personally meaningful projects while exploring deep ideas from a particular domain of knowledge. While Logo's central domain was mathematics, children were also learning how to program and, most importantly, how to think in new ways when making their own projects.

Coding playgrounds extend the notion of microworlds by making explicit the connection to playfulness and the multifaceted dimensions of human development. Going beyond the cognitive is important when addressing early childhood (four to seven years old). This is a life stage characterized by genuine curiosity and desire for learning about many things. Children need to learn about the natural world and the artificial world, the world of emotions and the world of ideas, the world by themselves, and the world of others in social contexts. This learning happens not only by thinking but by doing in a developmentally appropriate way (Bredekamp, Copple, & National Association for the Education of Young Children, 1997). Thus, coding playgrounds, or microworlds, must support experiences that engage young children in positive behaviors within the full range of human experience.

POSITIVE TECHNOLOGICAL DEVELOPMENT

Grounding my work in constructionism, I developed a framework called Positive Technological Development (PTD) (Bers, 2012) to highlight how technologies can engage children not only in thinking in new ways but also in behaving in new ways. The PTD framework identifies six positive behaviors that coding playgrounds or microworlds should support: communication, collaboration, community-building, content creation, creativity, and choices of conduct. From a theoretical perspective, these behaviors are associated with personal assets that have been described by decades of research on positive youth development as needed for thriving in life (Lerner, Almerigi, Theokas, & Lerner, 2005).

PTD is a natural extension of constructionism, but it explicitly incorporates psychosocial, civic, and ethical components. The framework examines the developmental tasks of a child growing up in our digital era and provides a model for designing and evaluating technology-rich youth programs (Bers, 2012). In learning experiences designed with a PTD framework, context plays a big role. It is not enough to use a wonderful coding playground or microworld. Classroom culture, curriculum, logistical, and physical organization of the learning environments, teachers, and more are as important as the technology itself. However, technologies must be designed in such a way as to support the emergence of positive behaviors in a developmentally appropriate way. ScratchJr and KIBO robotics are two examples of those technologies.

COMPUTATIONAL PLAYGROUNDS FOR EARLY CHILDHOOD

Papert described microworlds as being "sufficiently bounded and transparent for constructive exploration and yet sufficiently rich for significant discovery" (p. 208). ScratchJr and KIBO provide both the bounds, grounded on developmental theory, and the richness of programming environments so that young children can create their own projects to communicate ideas and express who they are.

Most importantly, within both of these playgrounds, coding—or the manipulation of the symbol system—is a major component of the experience, but not the only one. Just like at the playground, children have many options for things to do. At the playground, children can go to the sandbox, the swing, the slide, or just run around. They can play with sticks, ride their bikes, or create fantasy worlds. Similarly, while using KIBO or ScratchJr, children can engage in all kinds of activities beyond coding. For example, in ScratchJr they can create and modify characters in the paint editor, record their own voices and sounds, and even insert photos of themselves that they take in the paint editor using the camera option.

ScratchJr is a free introductory programming environment for young children ages five to seven. Inspired by Scratch (Resnick et al., 2009), ScratchJr was first launched as a freely downloadable app on iPads in July 2014 and has since been released for multiple platforms (Bers & Resnick, 2015). Used in classrooms and homes worldwide, ScratchJr enables children to create interactive stories and games by snapping together graphical programming blocks to make characters move, jump, dance, and sing, without the need of knowing how to read or write.

ScratchJr has a small basic set of graphics compared to the hundreds available in Scratch. Just as playgrounds for younger children offer limited play structures when compared to those for older children, this design decision was motivated by our overarching theme that "less is more" to ease children's difficulty in navigating vast arrays of options. However, it provides tools for children to create their own new graphics.

Whereas ScratchJr provides a screen-based coding playground, KIBO robotics offers a tangible experience (Bers, 2018). Children can build their own robot, program it to do what they want, and decorate it with art supplies. KIBO gives children the chance to make their ideas physical and tangible—without requiring screen time from PCs, tablets, or smartphones. As a playground, KIBO offers motors, lights, sensors, and art platforms as well as wooden blocks to sequence commands for the robot to follow once it is programmed.

The aesthetic features of KIBO, with an "unfinished" look, invites children to complete the robot using their own imaginative creations. Much like a blank canvas or unsculpted clay, KIBO inspires children to add to it. This supports a variety of sensory and aesthetic experiences. KIBO supports children in making almost anything: a character from a story, a carousel, a dancer, a dog sled. The possibilities are endless, as wide as children's own imaginations. The physicality of KIBO invites children to use their bodies, like in the playground, and engage in motor activities, while also collaborating with others.

Both KIBO and ScratchJr are coding playgrounds that support different dimensions of the multifaceted process of learning. Although we could claim that they are also microworlds, in Papert's sense, the term *coding playgrounds* makes explicit the emphasis on playfulness and the developmental needs of young children.

CONCLUSION

We do not always take children to the playground. There are other places to visit and other skills to develop. But when we do go to the playground, we want it to be a developmentally appropriate space. The same applies to programming environments for young children. There are other kinds of technological environments to explore, games and apps, simulations, and social media. But when young children are exposed to coding, playgrounds, and not only microworlds, provide a powerful metaphor for the best kind of learning experiences for young children. These playgrounds go beyond the cognitive dimension to encompass the social, emotional, personal, and moral dimensions.

ACKNOWLEDGMENTS

This work is possible through generous funding from the National Science Foundation (NSF) and the Scratch Foundation. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the author and do not necessarily reflect the views of neither NSF nor Tufts University. ScratchJr was developed as a collaboration between the DevTech research group at Tufts University, the MIT Lifelong Kindergarten Group, and the Playful Invention Company, with funding from the NSF and the Scratch Foundation. KIBO was developed by the DevTech research group in 2011 through generous funding from the NSF and is being commercialized worldwide by KinderLab Robotics.

REFERENCES

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

Bers, M. U. (2018). *Coding as a playground: Computational thinking and programming in early childhood. London, UK:* Routledge.

Bers, M. U., & Resnick, M. (2015). *The official ScratchJr book*. San Francisco, CA: No Starch Press.

Bredekamp, S., Copple, C., & National Association for the Education of Young Children. (1997). *Developmentally appropriate practice in early childhood programs*. Washington, DC: National Association for the Education of Young Children.

Lerner, R. M., Almerigi, J. B., Theokas, C., & Lerner, J. V. (2005). Positive youth development: A view of the issues. *The Journal of Early Adolescence*, *25*(1), 10–16.

Papert, S. (1980). Computer-based microworlds as incubators for powerful ideas. In R. Taylor (Ed.), *The computer in the school: Tutor, tool, tutee* (pp. 203–210). New York, NY: Teacher's College Press.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books, Inc.

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

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–36.