JRTE | Vol. 46, No. 1, pp. 81–101 | ©2013 ISTE | iste.org/jrte

Let's Dance the "Robot Hokey-Pokey!": Children's Programming Approaches and Achievement throughout Early Cognitive Development

Louise P. Flannery & Marina Umaschi Bers
Tufts University

Abstract

Young learners today generate, express, and interact with sophisticated ideas using a range of digital tools to explore interactive stories, animations, computer games, and robotics. In recent years, new developmentally appropriate robotics kits have been entering early childhood classrooms. This paper presents a retrospective analysis of one study within a design-based robotics research program. We examine how patterns of cognition throughout early childhood relate to programming approaches and achievement in a robotics context. The findings lay a foundation for applying cognitive developmental theory to early technology education and inform the evaluation of the study's programming and robotics technologies and curriculum. (Keywords: cognitive development, robotics, computer programming, early childhood education, STEM)

Children today, as in the past, explore and create artifacts from the materials around them. Clearly, though, the affordances of those materials have dramatically changed with the proliferation and evolution of computers. Activities from games to model-making and storytelling that first took form in the physical world have migrated into digital territory (Bers, 2008; Resnick et al., 2009). Young children's computer use need not be synonymous with passivity or pushing down content and learning strategies better suited to older learners (National Association for the Education of Young Children [NAEYC] & Fred Rogers Center, 2012). Novel advances in human-computer interaction mechanisms let young children engage in digital creation, including by programming child-friendly robots to carry out sequences of behaviors and interactions (e.g., Bers & Horn, 2010).

Working with age-appropriate programming tools and curricula, children can creatively problem solve and explore powerful interdisciplinary skills and knowledge (Bers, 2010; Bers & Horn, 2010; Grover & Pea, 2013). However, as new technological tools reach ever-younger children, more research is needed to define expectations for developmentally appropriate technology features and learning activities. The TangibleK Robotics Project (http://ase.tufts.edu/devtech/tangiblek/), an ongoing design-based research initiative conducted by the DevTech Research Group at Tufts University, explores learning trajectories

and developmentally appropriate technical and curricular designs for early-childhood robotics. Unanticipated observations in one study—namely the wide variety in young children's abilities to attend to and solve introductory programming and robotics challenges—raised questions about the origin of the disparities. The purpose of this article is to explore a preliminary hypothesis that cognitive development, which drives dramatic changes in reasoning throughout early childhood, might also influence achievement in learning to program robots. In addition to connecting classical developmental theory with an emerging domain, the overarching goal of this analysis is to evaluate the developmental assumptions underlying an existing programming tool and curriculum.

Programming in Early Childhood

During the 1970s and 1980s, the first personal computers introduced Logo, a text-based programming language (Logo Foundation, 2000). Meanwhile in a research lab, the first tangible children's programming language also debuted (Perlman, 1976). Since then, theoretical and empirical work has aimed to clarify the possible benefits of children's learning to program and the features of technologies and instruction that support positive outcomes (e.g., Battista & Clements, 1986; Bers, 2008; Clements & Gullo, 1984; Farr, Yuill, & Raffle, 2010; Horn, Solovey, & Jacob, 2008; Liao & Bright, 1991; Marshall, 2007; Pea & Kurland, 1984; Pea, Kurland, & Hawkins, 1985). Deep and meaningful learning can occur when children have rich problems to solve and powerful tools to work with (Martin, Mikhak, Resnick, Silverman, & Berg, 2000; Papert, 1993). Re-programmable—rather than minimally interactive pre-programmed—toys (Bergen, 2001) give children ownership over a piece of their world as they playfully imagine, create and program, explore, share, and reflect on their efforts (Bers, 2012; Resnick, 2006, 2007).

Our perspective on programming in early childhood education encompasses two bodies of theoretical work: computational thinking, which addresses problem solving with computers; and technological literacy and fluency, which examine expressivity with new technologies. More specifically, computational thinking describes a broad and still-coalescing range of analytic and problem-solving skills, dispositions, habits, and approaches as applied to solving problems with computers and algorithms (Barr, Harrison, & Conery, 2011; Barr & Stephenson, 2011; Grover & Pea, 2013; Lee et al., 2011; Guzdial, 2008; International Society for Technology Education [ISTE] & Computer Science Teachers Association, 2011; Wing, 2008). Building and programming computational artifacts can facilitate children's engagement in such high-level cognitive processes as creative design, problem solving, divergent thinking, and reflectivity, given a learning context that fosters active, iterative, and retrospective thinking and the appropriation of "failures" as intermediate learning opportunities en route to success (Clements & Meredith, 1992; Papert, 1993; Resnick, 2006).

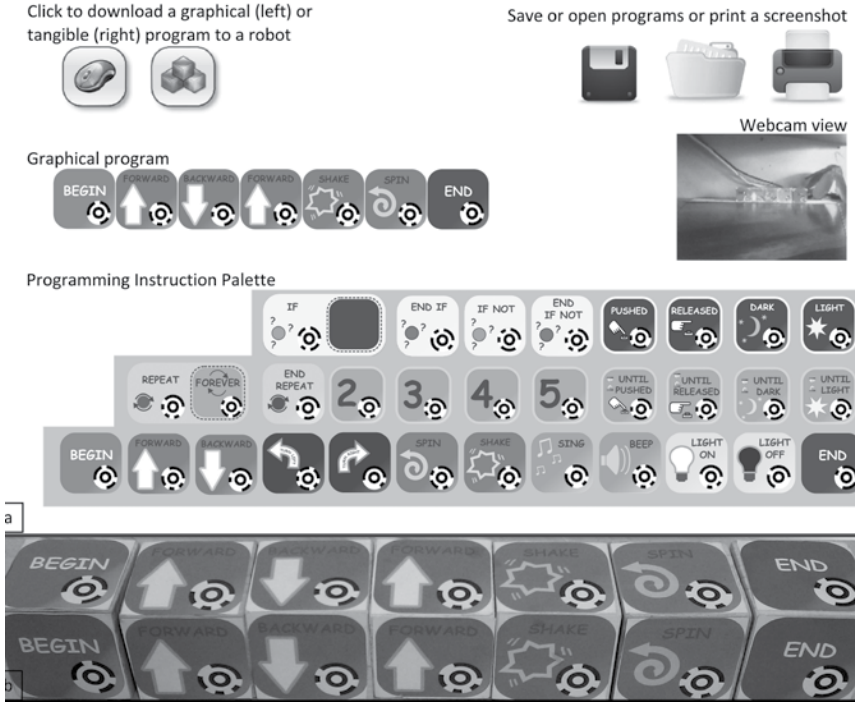

Figure 1. On-screen features of the CHERP interface, including a "Hokey-Pokey" solution (1a), and the tangible interface, with interconnecting wooden blocks (1b).

Frameworks describing new media literacies and technological fluency use different terms to do so, but they all focus beyond procedural knowledge of using computers for word processing or internet research (Massachusetts Department of Education [MA DOE], 2006; 2008; McGill & Volet, 1997) and toward creating content as well as exploring and communicating ideas in a variety of computer-based media (Buckingham, 2007; Jenkins, Purushotma, Weigel, Clinton, & Robison, 2009; New Media Consortium, 2005; Papert, 1993; Peppler & Kafai, 2007). Just as literacy education aims beyond decoding letters, words, and sentences toward oral and written fluency, so too should children develop technological fluency or expressivity.

Technical and interface design advances can and have led to increasingly child-friendly tools that support engagement with such learning—including in early childhood (Grover & Pea, 2013; Revelle, 2013). Children in the TangibleK Robotics Project used a programming language called CHERP (see Figure 1), which is designed specifically for their age range to create behaviors for robots they build from LEGO robotics construction kits (Bers, 2010; DevTech Research Group, 2011). Intended for programming robotic vehicles with such high-level instructions as "Forward" and "Beep," CHERP offers children the flexibility to switch between on-screen and tangible interfaces as they see fit (Horn, Crouser, & Bers, 2011), depending on manual dexterity,

perceived social appeal of the interface, and the challenge of the programming task at hand. The programming instructions, their representations, and the hybrid interface for manipulating them are cognitively and physically accessible to kindergarteners (Bers & Horn, 2010). Children can use CHERP to explore powerful ideas from technology-based domains that are often and unnecessarily reserved for older children or adults (Bers, 2008; Bers & Horn, 2010).

Given such early childhood educational technologies, it is up to society to understand how to use them to foster positive learning outcomes through formal and informal curricular designs. For instance, explicit instruction supports learning transfer of domain-specific, technological, and metacognitive knowledge from technology-based learning to other domains (Clements & Gullo, 1984; Clements & Meredith, 1992; Klahr & Carver, 1988; Nir-Gal & Klein, 2004; Salomon & Perkins, 1987). A less researched component involves understanding how children's cognitive developmental level affects their technology use. Application of this knowledge can result in effective pedagogical designs to teach not only technology-specific content but also the high-level thinking skills that make up computational thinking and technological fluency.

A Contemporary Revision of Piaget's Stages

Though computers existed only late in Jean Piaget's life, his work on developmental patterns in children's thinking applies to young children's problem-solving-based computer use today. Core tenets of Piaget's field-transforming theory remain salient to current understandings of cognitive development, from the active and internally driven construction of knowledge to the coherent patterns of cognition seen at different ages (Flavell, 1996). Over time, scholars have also extensively reshaped his theories to specify mechanisms by which stage transitions occur and new cognitive structures arise, and to address individual differences observed in these transitions (e.g., Case, 1984; Feldman, 2004; Fischer, 1980; Granott & Parziale, 2002; Kuhn, 1995; Lewis, 2000; Siegler & Crowley, 1991; van Geert, 1998).

Feldman (2004) reconciles Piaget's theory with empirical observations of uneven and gradual progress from one stage to the next and incorporates the principle of emergence common among other neo-Piagetian theories. Underpinning this analysis, a key insight of Feldman's (2004) theory is that, halfway into each stage, children move from actively constructing new systems of thought to energetic application of those systems, building a bridge toward upcoming cognitive structures. Children in the present study ranged from about 4.5 to 6.5 years old, a timespan during which children experience substantial cognitive growth (Case, 1984) as they move from late pre-operations, somtimes called intuitive operations, toward early concrete operations. The characteristics of Feldman's phases highlight the salience of many developmentally influenced thought processes to reasoning, problem solving, computer programming, and understanding how digital objects work.

During the late pre-operational stage of cognitive development (ages 4–6), children extend and apply culturally learned symbol systems to interactions with the physical and social world and rely on transductive reasoning—the joining of unrelated observations in idiosyncratic ways that the child finds immensely compelling, although they defy adult logic (Feldman, 2004; Gardner, Kornhaber, & Wake, 1996; McDevitt & Ormrod, 2002). These cognitive structures form the precursors to subsequent increasingly adult-like logic (Feldman, 2004; Gardner et al., 1996). Also shaping the cognition of children in this stage are difficulties relating multiple aspects of objects, distinguishing appearances from reality, taking multiple psychological and physical perspectives, and mentally manipulating objects (Feldman, 2004; Gardner et al., 1996; McDevitt & Ormrod, 2002). While the 4- or 5-year old child is certainly hard at work asking questions and theorizing about how the world works, the cognitive tools s/he employs to construct those theories are unique to that age (Feldman, 2004); thus the child's conclusions also differ drastically from those of older children and adults.

The transitional period described by Feldman's (2004) model is characterized by the staggered appearance of concrete operational mental processes and inconsistent use of these strategies. The child becomes interested in exploring concepts and cognitive processes that previously seemed irrelevant: using symbol systems to begin constructing categories, hierarchies, and other relationships, based on the child's own experience (Feldman, 2004; Gardner et al., 1996). During this period, children may switch between patterns of thought characteristic of the developmental level they are leaving and the level they are entering, as the grasp of these concepts will solidify only later on (Feldman, 2004).

By around 6 years of age, a child is likely switching into concrete operational patterns of thinking (Lightfoot, Cole, & Cole, 2009). S/he is now familiar with representing the world with both mental and physical symbols, and the work of the second phase of pre-operations has prepared the child's cognitive structures for transformation toward interest and competence in more logical reasoning (Feldman, 2004). Children this age begin to simultaneously consider multiple aspects of situations or objects, allowing them to work with conservation of quantities and the concepts of categories and hierarchies (Feldman, 2004). A 6-year-old increasingly understands physical and psychological points of view other than his/her own. Around this time, a child also relies increasingly on logical reasoning about causal relationships, empirical observations, and the distinction between appearances and reality (Lightfoot et al., 2009; McDevitt & Ormrod, 2002). While a child this age typically uses concrete materials to build mental models or representations (Feldman, 2004), s/he does not rely on the physical object's immediate presence to mentally consider and manipulate it (Lightfoot et al., 2009). The 6-year-old also becomes more able to plan a series of actions to fulfill a goal and to think flexibly in doing so, and cognition in this stage is aided

by increasing memory capacity and meta-cognition (Lightfoot et al., 2009). Over the next 2 or 3 years, the child will consolidate the cognitive structures necessary to successfully apply these new cognitive skills (Feldman, 2004).

Understanding the natural distinctions in cognitive resources available to a 4-year-old and a 6-year-old, as described by Piaget and subsequent cognitive developmentalists, may aid in evaluating technology-based materials and activities that engage children in logical reasoning—such as programming a robot.

Research Design

The present analysis focuses on one iteration of the TangibleK Robotics Project, a design-based research program (e.g., Barab & Squire, 2004; Cobb, Confrey, diSessa, Lehrer, & Schauble, 2003) intended to detail what kindergarteners can understand about programming and robotics and how developmentally appropriate tools can support learning in early childhood classrooms. The study described in this paper occurred during TangibleK's third year, following piloting of the CHERP programming technology and a classroom curriculum. In depth and laboratory based, this study documented individual children's learning and problem-solving processes. Participants of this study attended a small-group session for pre-assessments and introduction to the technologies, and then three individual sessions in which they constructed a robotic vehicle, learned new programming concepts, attempted a programming challenge, and reflected on their work. Post-assessments took place during the final session. We present relevant data here to examine the role of cognitive development and subsequently inform evaluation of the programming and robotics materials and curriculum.

Throughout this study, researchers observed an unanticipated range in children's uses of the programming tools and their programming achievement. This article, framed as a retrospective analysis, examines children's first individual programming activity to uncover what patterns exist in programming approaches and achievement based on estimated level of cognitive development.

Variables

Cognitive developmental level. Cognitive development was not among the original study variables; however, we designed a framework to classify each child's thinking processes during the programming task, as representative of one of three substages of cognitive development from Feldman's (2004) revision of Piaget's model. Programming, while not a traditional Piagetian assessment, is a rich source of logico-mathematical and deductive reasoning—the types of thinking most relevant to Piaget's theory (Case, 1984). We compiled characteristics of the framework's three categories—late pre-operations, transitional, and early concrete operations—from the literature and mapped

Table 1. Cognitive Stage Markers in Programming Rubric

	Pre-Operational	Transitional	Concrete Operational
Goal Orientation	Focuses primarily or exclusively on open-ended exploration. May try the "Hokey-Pokey" (HP) nominally or cursorily. *	Tries HP (with interest and effort) but leaves it due to interest in other explorations or being unable to debug further (may claim an incomplete program is successful).	Focuses primarily on HP with little or no redirection through to a nearly or fully complete solution. May explore openly before/during HP.
Initial Solution	Nominal, cursory, or no attempt. OR Intuitive approach (considers actions but not order).	Intuitive approach with limited systematic logic (order).	Logical approach (step-by-step sequencing).
Debugging Attitudes and Strategies	Indifferent to the need to debug or to the results of any unsuccessful efforts.	Interested in improving the program but cannot figure out how.	Driven to find best answer. OR Gets answer right away and knows it.
	Nominal, cursory, or no attempt. OR Intuitive approach (e.g. guess-and-check).	Mixed approach intuitive / logical & empirical. Limited / inflexible ideas on how to systematically debug.	Logical / empirical approach. Flexible if one idea does not work.
Perspective and Classification	Attributes agency inappropriately to self versus the robot.		Attributes agency appropriately to self versus the robot.
	Confused by different orientations of the computer, blocks, robot, map, and self.		Unconstrained by different orientations of the computer, blocks, robot, map, and self.
	Single classification for "blocks."		Multiple classifications for "blocks."

**This includes situations in which the child verbally claims to be working on the Hokey-Pokey, perhaps in an effort to avoid conflict with a perceived authority figure, but actually makes a completely unrelated program and shows through other behavior or speech that the Hokey-Pokey is not the actual goal.*

them to a programming context. We gave three core elements of the children's approach subscores based on video and written data: the child's goal, and the intuitive versus systematic nature of the initial solution strategy and solution revision strategies. We determined overall developmental level from the component scores, taking into consideration other cognitive operations as needed (see Table 1).

Programming achievement. We assessed children's programming achievement based on their ability to program a mobile robot to dance the "Robot Hokey-Pokey":

> You put your robot in (Forward)/You put your robot out (Backward)/ You put your robot in (Forward)/And you shake it all about! (Shake)/ It does the Hokey-Pokey, and it turns itself around (Spin)/ And that's what it's all about! (Sing)/*Clap, clap!* (Beep, Beep)

After the researcher ensured familiarity with the song and its actions, the child worked on the challenge without conceptual assistance from the researcher. Instead, the researcher reflected back any questions, creating a supportive environment without providing answers. After a set time, the researcher helped the child complete any unfinished aspects of the challenge.

Table 2. "Hokey-Pokey" Program Completeness Assessment Rubric

Scoring Instructions	
• Must represent Forward | Backward | Forward | Shake | Spin. Do not score Begin/End. • May use Turn(s) for Spin, sounds at the beginning or end, or a second, consecutive Shake or Spin. • Use the fewest possible fixes to reach an accepted solution.	
Definitions of Fixes	
Addition	One of the 5 basic solution instructions is missing and needs to be added.
Swap	2 consecutive instructions need to be switched.
Deletion	An instruction needs to be removed.
Distinguish	The child consistently confused 2 similar instructions (e.g. Backward and Forward), so these instructions need to be exchanged.
Scale and Examples	
4 – no fixes	
3 – one fix	1 addition Forward | Backward |Shake | Spin 1 swap Forward | Forward | Backward |Shake | Spin 1 deletion Forward | Backward | Forward | Shake | Backward | Spin
2 – two fixes	2 additions Forward | Backward | Shake 2 swaps Forward | Forward | Backward | Spin | Shake 2 deletions Forward | Backward | Shake | Forward | Shake | Spin | Right 1 addition, 1 swap Backward | Forward | Spin | Shake 1 addition, 1 deletion Forward | Sing | Backward | Forward | Shake 1 swap, 1 deletion Backward | Forward | Forward | Shake | Left | Spin
1 – three+ fixes	The child clearly attempted to make a Hokey-Pokey program. The program has 2 correct actions and few if any extras, other than reduplications. These tend to be fairly short. 2 instructions, no reduplications Forward |Shake | Sing 2 instructions, with reduplications Backward | Backward | Sing | Sing | Spin | Spin A copy of CHERP's graphical instruction set, given the child's recognition that parts match the song.
0 – avoided task / unrecognizable	The child did not attempt to make a Hokey-Pokey program. OR The program is so incomplete as to be unrecognizable as a clear attempt at the Hokey-Pokey. (The child may or may not have claimed it to be a Hokey-Pokey attempt. These tend to be long compared to programs assessed as 1.)

We assessed programming achievement at two levels. Defined as purposefully choosing programming instructions to match the song's actions, *correspondence* was assessed on a Likert scale from 0 (cannot achieve) through 5 (achieves without assistance); intermediary values represented increasing levels of support that the child needed to successfully apply this cognitive skill within the timespan of the session. *Program completeness* measured correctly sequencing instructions in addition to making accurate action-instruction correspondences. The assessment scale, shown in Table 2, indicates how many changes a child's independently created program needs to match an accepted solution; the scale ranges from 0 (unrecognizable as an attempt at the "Hokey-Pokey") to 4 (all required instructions in exact order).

Sample

The original study included 36 children recruited from Boston-area towns and cities via the research group's website, email, and the snowball effect. Twenty-nine met inclusion criteria for this analysis: sufficient documentation of the

child's thinking (as communicated verbally or behaviorally) to allow inter-rater reliability testing, the child's attempt of the assigned programming task or a structural equivalent, and no evidence that the study format prevented the child from demonstrating his or her best efforts. The 29-child subsample was composed of 38% girls and 62% boys. Kindergarteners (20 children) made up 69% of the sample, and preschoolers (9 children) made up the remaining 31%. Age at the time of each child's first session ranged from 4.4 years to 6.6 years; mean age was 5.6 years. The group came about evenly from urban and suburban neighborhoods. Eleven children (38%) attended public schools, whereas 18 (62%) attended private schools. (The lower availability of public preschools compared to public kindergartens probably accounts for the skew.) Children also came from multiple ethnic, cultural, and language backgrounds.

Most families (89%) had a parent with a master's or doctoral degree. More than half had a parent whose culminating degree and/or occupation related to science, technology, engineering, or math. Almost a third of the children had a parent with prior exposure to robotics, and two-thirds had a parent with prior programming experience. Three-quarters of the children used a computer at home, mostly to play games. A third had used programmable robots, but no parents reported that their children had programmed (there may be discrepancies in definitions of "programming"). Although this sample's characteristics may reflect the subset of families interested in early childhood robotics more than the general population of U.S. families with preschoolers and kindergarteners, it does provide a useful starting point for exploratory research.

Results

Comparison of Developmental Levels

Using the cognitive development measure described earlier, we categorized each child's reasoning as being characteristic of late pre-operational (28% of the sample), transitional (24%), or early concrete operational (48%). Inter-scorer reliability tests showed precise agreement (two items; $\alpha = 1.00$). The skew toward concrete operational reflects preferential recruitment of kindergarteners for the study's original goals. Each developmental category included roughly the same proportions of demographic, experiential, and parental characteristics as the overall study sample, with a few variations.

Children in each successive category were older, on average ($F(2,28) = 5.5$, $p < .01$). The mean age of pre-operational children fell at 5.2 years ($SD = 0.50$); almost half a year higher for the transitional category, 5.6 years ($SD = 0.69$); and highest for children in concrete operations, 5.9 years ($SD = 0.40$). Only the difference between the earliest and latest developmental categories was statistically significant ($p < .01$), reflecting broad individual variability in the age at which developmental transitions occur. Other

differences were artifacts of the sample. Children in the earliest cognitive developmental phase were more likely to attend private school ($\chi^2(2, N = 29) = 7.16, p < 0.05$), use computers at home ($\chi^2(2, N = 29) = 9.89, p < .01$), and have a parent with prior programming experience ($\chi^2(2, N = 28) = 6.71, p < .05$) than children in the higher levels. Overall, aside from expected age differences, the three developmental groups had characteristics that were fairly similar to the sample as a whole.

Programming in Late Pre-Operations

Eight children (28% of the sample) exhibited late pre-operational reasoning during the "Hokey-Pokey" activity. Half of these children disregarded the given challenge to focus instead on open-ended explorations of the robot's capabilities. Two children claimed that exploratory programs matched the "Hokey-Pokey," perhaps to appear compliant. The other half of the children in this category tried, at least for a while, to create the "Hokey-Pokey." Relying heavily on trial-and-error, they thought of one or two actions, with little further progress. They had difficulty assessing whether the robot had done what they expected and what to fix if it had not. For some, starting over seemed easier than revising their in-progress solution. Having exhausted their intuitive strategies, children waited for researcher assistance; turned to more familiar activities, such as building with LEGOs; or explored the programming tool more. The lack of progress toward the "Hokey-Pokey" goal concerned some children, while others happily ignored the unfinished task in favor of their general exploration.

Programming in the Transitional Phase

Seven children (24% of the sample) exhibited a mix of characteristics of pre-operations and concrete operations, placing them in this intermediate category. All of the children were interested in the given activity and made some systematic progress toward a "Hokey-Pokey" program. However, they encountered difficulties in fully applying systematic and empirical strategies. The abilities and general approaches seen within this group varied much more than in the other groups. It is likely that this category comprises children who were just beginning to use some concrete operational structures as well as some children who were much further along in the transition.

Four transitional children started with an intuitive guess and were able to make some systematic revisions before getting stuck. They ended up with either a program that nearly matched the "Hokey-Pokey" or a program that matched the song in length and a few specific actions. One child knew what actions she needed—and in the correct order—but was unable to interpret CHERP's "Forward" and "Backward" instructions as representations of the "in" and "out" movements in the verse. Two other children began with a systematic approach. One decided his program was close enough and declined to try improving it; the other systematically improved his program—with

encouragement—before resorting to a guess-and-check strategy and ending up with an almost complete solution.

Children in the transitional group each had difficulty with at least one aspect of debugging: recognizing a problem with the current solution, generating a hypothesis about the cause, and attempting to solve the problem. In response to this situation, the children variously insisted that they were stuck, reverted to unsuccessful intuitive strategies, or decided that an incomplete solution was satisfactory. Sometimes children applied systematic debugging toward interesting problem variations. One child worked hard to match his program to the song's length even though the specific actions did not match, and another tried to direct the order and timing of lines as the researcher sang them so that the song would match his program rather than the other way around.

There were two categories of final program completeness within the transitional group. Half the children ended up with programs resembling "Hokey-Pokey" prototypes using only two of the five actions, and the children were quite aware that their programs were incomplete. The other half ended up with programs with only one instruction off, but they seemed to have no interest in or awareness of this difference. These final programs are almost on par with those of children in the concrete operational group. However, the transitional children who made them tended not to have done any debugging, which requires additional cognitive flexibility and was common to concrete operational children. Additionally, transitional children who used musical instructions tended to do so simply to indicate that the "Hokey-Pokey" is a song, rather than to represent the last line of the song and the clapping that follows. Overall, there was an interesting bimodal pattern of reasoning among children in the transitional category as well as variety in the unique mixes of systematic and intuitive strategies used by each child.

Programming in Early Concrete Operations

Fourteen children (48% of the sample) used cognitive strategies typical of early concrete operations. Their work during the "Hokey-Pokey" programming activity is markedly different from that of the pre-operational children and, to a somewhat lesser extent, from that of the transitional children. Children scored as concrete operational tended to stay on task once they started, staying focused until arriving at a complete or nearly complete solution. The children in this developmental category used systematic approaches to create and tweak their programs and were quick to notice errors and try to fix them, a striking and significant difference from children in the other categories. To create their first programs, children in concrete operations relied more on parsing the song step by step than on intuitively recalling its elements. Then they applied their empirical observations of the robot carrying out the program to assess their solution. Some also "read through"

Table 3. Programming Achievement by Cognitive Developmental Level

Outcome Variable	Level	n	M	SD
Correspondence	Full Sample	29	3.86	1.66
	Pre-Operational	8	1.87	1.46
	Transitional	7	3.86	1.46
	Concrete Operational	14	5.00	0.00
Program Completeness	Full Sample	29	2.31	1.69
	Pre-Operational	8	0.13	0.35
	Transitional	7	1.86	1.07
	Concrete Operational	14	3.79	0.43

Note. Interscorer reliability was very high on both measures (two items; $\alpha = .99$ in each case).

their programs block by block while saying the song to themselves to assess whether the program and song matched.

Children in concrete operations recognized whether an instruction was missing, unnecessary, or out of order, which children in the pre-operational category had great difficulty doing. Children in this category were rarely satisfied with a program that was not entirely complete, as opposed to children in the transitional and pre-operational categories, who were quite likely to declare a partially complete or even completely unrelated program successful. This evolution of self-imposed standards seems to parallel the increasing orientation toward exactness in "bring[ing] a productive situation to completion" during the elementary school years (Erikson, 1998, p. 72).

Three of the 14 concrete operational children put together a correct solution on their first try. The others revised and tested (debugged) their programs one to five times. Of the 11 children who used such an iterative trouble-shooting process, about half began with a long program (four to seven instructions), which they corrected by re-ordering, adding, or removing instructions as needed. The other half began with a small portion of the final solution—two or three instructions—and built up to the final solution with each debugging iteration.

Developmental Level and Achievement

Children at each level of cognitive development varied distinctly in their approaches to programming. This trend is also seen in their achievement of core programming concepts and skills. Average achievement levels of two programming concepts—correspondence and final program completeness—were high across the full sample, but the different distribution of scores in each developmental category warrants further discussion.

For making correspondences between programming instructions and robotic actions, the overall mean score was nearly 4 out of 5—needing little help to apply this concept (see Table 3). However, 62% of the children achieved the highest score (a ceiling effect), whereas the remaining 38% were distributed among the lower scores (see Figure 2). The distribution of

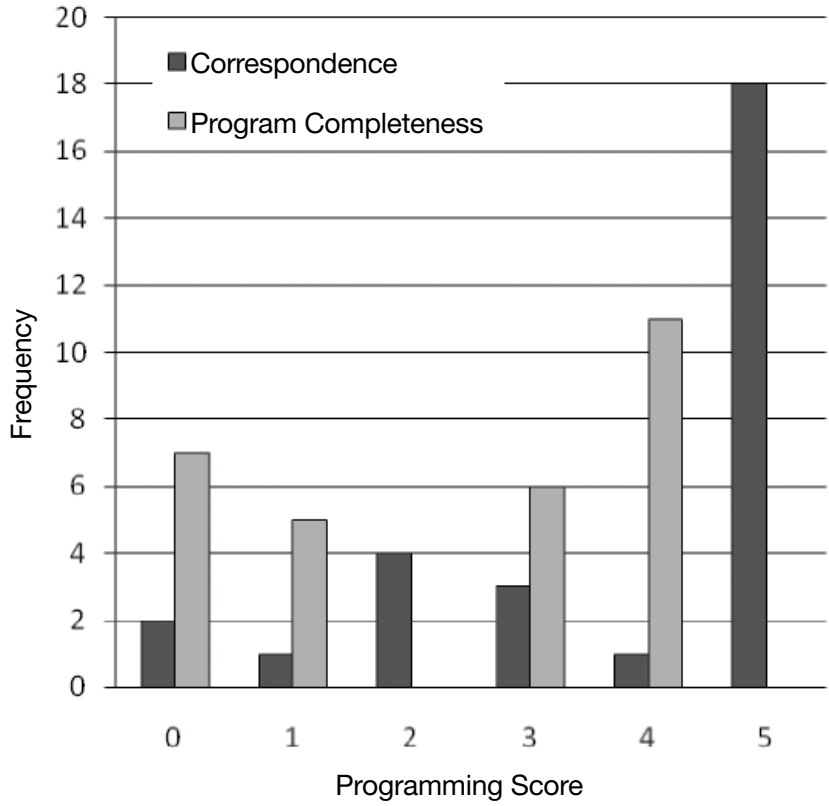

Figure 2. Frequency of programming achievement scores within the sample. (Note: The program completeness scale ranged from 0 to 4, in contrast with the correspondence scale of 0–5.)

scores within each cognitive developmental category is shown in Figure 3 (p. 94). Pre-operational children scored lowest on average, generally needing periodic to step-by-step intervention to select instructions based on their actions (M = 1.87 out of 5, SD = 1.46). Their scores are distributed across nearly the full range of possible values, with a drop-off toward the top of the range. Children in the transitional category scored statistically significantly higher (M = 3.86, SD = 1.46, $p < .05$). Their scores are split between the middle and uppermost ranges of possible scores; they needed either periodic or no support with correspondence. Those in the concrete operational category (M = 5.00, SD = 0.00) also scored statistically significantly higher than the pre-operational group ($p < .001$), and their uniform achievement of the highest score indicated no need of help in applying this skill.

Looking at program completeness (how close the child's program was to using only the correct instructions in the correct order), the overall sample mean was less than 2.5 out of 4 (see Table 3). However, the scores followed

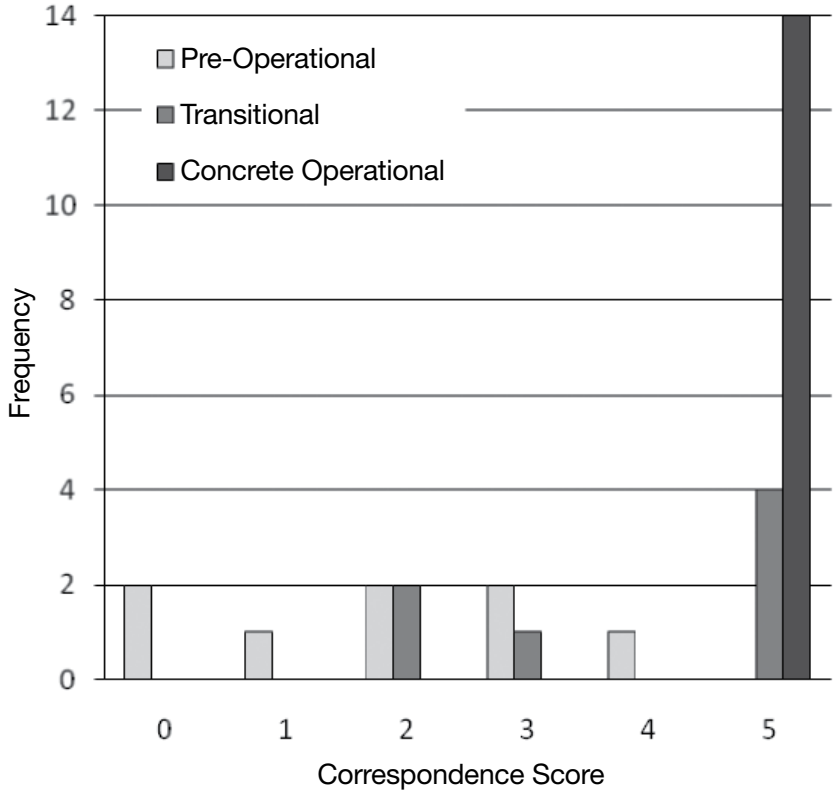

Figure 3. Frequency of correspondence scores within each cognitive developmental group.

a bimodal distribution (see Figure 2, p. 93). About 40% of children's final programs were either very rough attempts at the "Hokey-Pokey" or even unrecognizable as an attempt at the "Hokey-Pokey." The other 60% of programs had the five basic actions solution in order or needed to fix a single instruction. Interestingly, no children achieved a midrange score (for a final program on the right track but needing two changes to be correct).

Figure 4 shows the distribution of completeness scores by developmental category. Children in the pre-operational group again scored the lowest, with an average near 0 out of 4 possible points ($M = 0.13$, $SD = 0.35$), meaning that their programs were generally unrecognizable as attempts at the "Hokey-Pokey." Children in the transitional category scored statistically significantly higher ($M = 1.86$, $SD = 1.07$, $p < .001$). Although this average score implies that their programs required two to three fixes to be complete, these children actually made either a nearly complete program or a nearly unrecognizable attempt at the solution, with no children scoring in the middle. Children in the concrete operational group scored the highest, with at most one change needed to make their program complete ($M = 3.79$, $SD = 0.43$, $p < .001$ compared to both other groups).

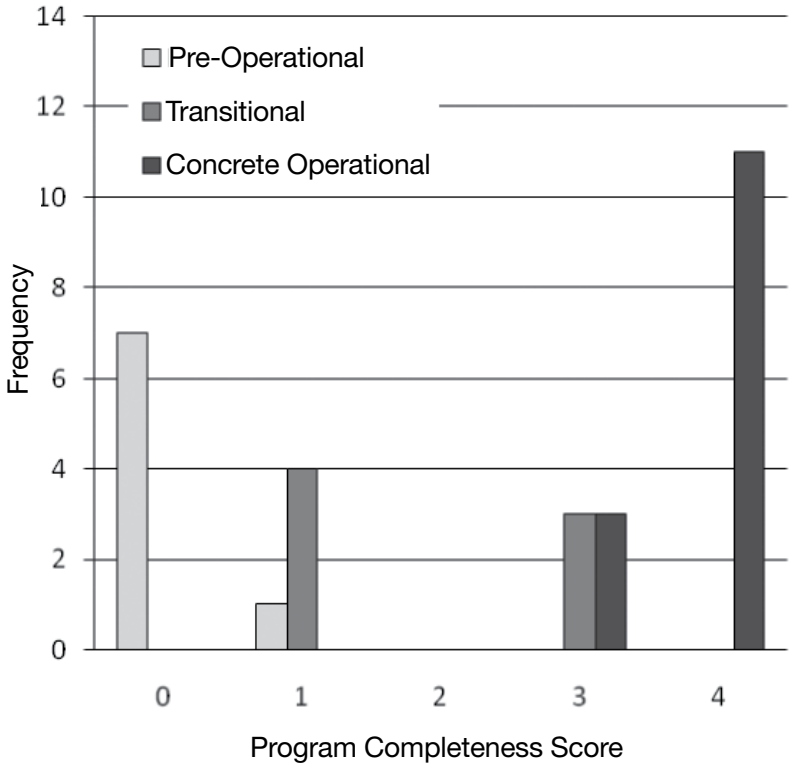

Figure 4. Frequency of program completeness scores within each cognitive developmental group.

Discussion

Several compelling results stem from this analysis. First is the preschool and kindergarten study participants' categorization by cognitive developmental markers seen in their thinking and problem-solving strategies as they programmed mobile robots. This categorization allowed a preliminary examination of the relationship of developmentally based cognitive characteristics with the programming skills young children can apply and the extent of their achievement. Children in the latter half of pre-operations tended to explore the possibilities and boundaries of CHERP rather than engage in specific, given challenges. The intuitive problem-solving strategies characteristic of this group made the "Hokey-Pokey" goal unattainable. Children in the first phase of concrete operations responded quite differently to the task. They enthusiastically generated iteratively more precise solutions. Unlike their pre-operational peers, they used empirical feedback and systematic logic to reach the goal. When taking on self-defined challenges, their goals were contextualized; they wanted to use CHERP to accomplish an imagined scenario and go beyond simply understanding how CHERP works, as younger children were satisfied to do. Results varied more within the group transitioning between cognitive

stages. These children were generally interested in solving the challenge, like the concrete operational group, and made some similar progress, though it was inconsistently systematic or empirically based. Like the pre-operational group, they became stuck before fully solving the challenge and often moved on to open-ended explorations.

Another interesting finding was that children across the age range of the study seemed to experience similar challenges and successes with manipulating the robotics and programming materials. However, children responded quite differently to the lesson goal and format. Given the evidence presented here, it is expected that children in different stages and substages of cognitive development would benefit from learning goals, activities, and scaffolding designed specifically for their distinct cognitive characteristics.

In the pre-operational stage of cognitive development, children's learning outcomes might improve given curricula focused on exploring the programming tool to discover its capabilities and boundaries and to begin to use it as an expressive medium. Over time, they can work toward solving short challenges, and they benefit from teacher interactions that scaffold careful observation and responding iteratively to the results of each new programming effort. These children can reason through or solve limited chunks of a larger programming challenge, but they need support in managing awareness of the scope of an activity like the "Hokey-Pokey" and the problem-solving process it requires.

Children who have entered the concrete operational stage, on the other hand, would likely benefit from an expanded curriculum of contextualized activities akin to the "Hokey-Pokey" task. Activities like this, whether defined by a teacher, a curriculum, or the child him/herself, provide a context for children in concrete operations to apply their growing systematic reasoning and meta-cognitive skills to increasingly complex activities and programming concepts.

To date, new implementations of the TangibleK curriculum have included several adaptations based on these ideas. For example, a slower pace and expanded focus on the introductory activities are used with the youngest participants, allowing more time to explore the tools; whole-group investigation of higher-level cognitive aspects of programming and robotics is also included. Expansion of the curriculum is also being tested with older children by providing multiple activities on each concept.

As this analysis was retrospective, and in many respects a pilot study, it had several areas warranting revision and follow-up. Although we grounded our method of estimating cognitive development in theory and literature, it can be expected that the cognitive and programming achievement variables are confounded to an unknown extent. For instance, pre-operational thinkers are likely to guess and check when problem solving, which naturally correlates to less effective correspondence and sequencing. New studies already in progress will employ distinct cognitive and programming measures to

allow correlational analysis and provide insights into revising the framework, perhaps by defining multiple transitional categories. A larger, more representative sample, including more girls and a wider range of parental backgrounds, would support the generalization of these findings. Analysis of multiple programming activities per child would also help paint a more detailed picture of children's programming.

Despite the limitations inherent in this retrospective analysis, its formulation provides a foundation for understanding the variability in programming tool use and achievement seen in this study, and for informing the evaluation of the TangibleK Robotics Project learning materials and goals. By analyzing the in-depth data collected from almost 30 preschool and kindergarten children, this study has connected a long-standing cognitive developmental theoretical tradition with the relatively newer realm of engaging young children with technologies that invite creativity and problem solving.

Conclusions

Children use new technologies from very young ages today as never before, but adults in their lives may not yet know how to support developmentally appropriate options, for instance, by considering the diverse cognitive characteristics children exhibit over the span of only a few years. This study, made possible by the design of a framework for retrospective estimation of cognitive development, can help parents, educators, and technology designers promote positive and meaningful learning experiences with new technologies. The results also point to the need for differentiated learning expectations and curricula for programming throughout the early childhood years and perhaps also the design of new programming and robotics technologies.

Some parents and teachers hesitate to have children engage in programming and robotics, citing a preference to limit "screen time." This well-founded concern highlights the need to differentiate the current vocabulary, as there is a vast difference in the cognitive activity fostered by screen-based activities for consumption (i.e., many video games and television) compared to those for production (i.e., programming and creative design). In using tools like CHERP, children move physically and cognitively between on- and off-screen materials as they imagine, plan, and construct a robot and its actions; they iteratively observe, analyze, and adapt their work amid new discoveries. In a classroom, children naturally investigate the work of their peers, collaborate, and negotiate over materials. The robot and its program serve as points from which to discuss and reflect on content even after the computer has been turned off.

Through rich processes of creation and problem solving, even young children can engage in programming robots' behaviors, bridging the physical and digital worlds, and actively exploring both general cognitive skills and domain-specific content in developmentally appropriate ways.

Acknowledgments

We would like to express our gratitude to Professors David Henry Feldman and Bakhtiar Mikhak. This work was supported by National Science Foundation Grant #DRL-0735657.

Author Notes

Louise P. Flannery is a research scientist and coordinator in the DevTech Research Group in the Eliot-Pearson Department of Child Development at Tufts University. Her research interests encompass cognitive development, reasoning, and learning throughout early childhood and the application of these to the design of new educational technologies, especially those for creative expression and construction. Please address correspondence regarding this article to Louise P. Flannery, Tufts University, Eliot-Pearson Dept. of Child Development, 105 College Ave, Medford, MA 02155. E-mail: louise.flannery@alumni.tufts.edu

Marina Umaschi Bers has joint appointments at Tufts University as professor in the Eliot-Pearson Department of Child Development and adjunct professor in the Computer Science Department. As head of the interdisciplinary DevTech Research Group, her research involves the design and study of innovative learning technologies to promote children's positive development.

References

Barab, S., & Squire, K. (2004). Design-based research: Putting a stake in the ground. *Journal of the Learning Sciences, 13*(1), 1–14. doi:10.1207/s15327809jls1301_1

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning and Leading with Technology, 38*(6), 20–23.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K–12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48–54. doi:10.1145/1929887.1929905

Battista, M. T., & Clements, D. H. (1986). Effects of Logo and CAI environments on cognition and creativity. *Journal of Educational Psychology, 78*(4), 309–318. doi:10.1037/0022-0663.78.4.309

Bergen, D. (2001). Learning in the robotic world: Active or reactive? *Childhood Education, 77*(4), 249–250.

Bers, M. (2008). *Blocks to robots: Learning with technology in the early childhood classroom.* New York: Teachers College.

Bers, M. (2010). The TangibleK Robotics Program: Applied computational thinking for young children. *Early Childhood Research & Practice, 12*(2). Retrieved from http://ecrp.uiuc.edu/v12n2/bers.html

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

Bers, M. U., & Horn, M. S. (2010). Tangible programming in early childhood: Revisiting developmental assumptions through new technologies. In I. R. Berson & M. J. Berson (Eds.), *High-tech tots* (pp. 49–70). Greenwich, CT: Information Age.

Buckingham, D. (2007). Digital media literacies: Rethinking media education in the age of the Internet. *Research in Comparative and International Education, 2*(1), 43–55. doi:10.2304/rcie.2007.2.1.43

Case, R. (1984). The process of stage transition: A neo-Piagetian view. In R. Sternberg (Ed.), *Mechanisms of cognitive development* (pp. 19–44). San Francisco: Freeman.

Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology, 76*(6), 1051–1058. doi:10.1037/0022-0663.76.6.1051

Clements, D. H., & Meredith, J. S. (1992). *Research on Logo: Effects and efficacy.* Retrieved from http://el.media.mit.edu/logo-foundation/pubs/papers/research_logo.html

Cobb, P., Confrey, J., diSessa, A., Lehrer, R., & Schauble, L. (2003). Design experiments in educational research. *Educational Researcher, 32*(1), 9–13. doi:10.3102/0013189X032001009

DevTech Research Group (2011). C.H.E.R.P. Retrieved from http://ase.tufts.edu/DevTech/tangiblek/research/CHERP.asp

Erikson, E. H. (1998). Eight stages of man. In C. L. Cooper & L. A. Pervin (Eds.), *Personality: Critical concepts in psychology* (pp. 67–77). London: Routledge.

Farr, W., Yuill, N., & Raffle, H. (2010). Social benefits of a tangible user interface for children with autistic spectrum conditions. *Autism, 14*(3), 237–252. doi:10.1177/1362361310363280

Feldman, D. H. (2004). Piaget's stages: The unfinished symphony of cognitive development. *New Ideas in Psychology, 22*, 175–231. doi:10.1016/j.newideapsych.2004.11.005

Fischer, K. (1980). A theory of cognitive development: The control and construction of hierarchies of skills. *Psychological Review, 87*(4), 477–531. doi:10.1037/0033-295X.87.6.477

Flavell, J. H. (1996). Piaget's legacy. *Psychological Science, 7*(4), 200–203. doi:10.1111/j.1467-9280.1996.tb00359.x

Gardner, H., Kornhaber, M. L., & Wake, W. K. (1996). *Intelligence: Multiple perspectives.* Fort Worth, TX: Harcourt Brace College.

Granott, N., & Parziale, J. (2002). Introduction. In N. Granott & J. Parziale (Eds.), *Microdevelopment: Transition processes in development and learning* (pp. 1–28). Cambridge, UK: Cambridge University Press.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43. doi:10.3102/0013189X12463051

Guzdial, M. (2008). Paving the way for computational thinking. *Communications of the ACM, 51*(8), 25–27. doi:10.1145/1378704.1378713

Horn, M. S., Crouser, R. J., & Bers, M. U. (2011). Tangible interaction and learning: The case a hybrid approach. *Personal and Ubiquitous Computing, Special Issue: Tangibles and Children.* doi:10.1007/s00779-011-0404-2

Horn, M. S., Solovey, E. T., & Jacob, R. J. (2008). Tangible programming and informal science learning: Making TUIs work for museums. In *Proceedings of the Seventh International Conference on Interaction Design and Children* (pp. 194–201). New York: ACM. doi:10.1145/1463689.1463756

ISTE & Computer Science Teachers Association (2011). *Operational definition of computational thinking for K–12 education.* Retrieved from http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf?sfvrsn=2

Jenkins, H., Purushotma, R., Weigel, M., Clinton, K., & Robison, A. J. (2009). *Confronting the challenges of participatory culture: Media education for the 21st century.* Cambridge, MA: MIT Press.

Klahr, D., & Carver, S. M. (1988). Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology, 20*(3), 362–404. doi:10.1016/0010-0285(88)90004-7

Kuhn, D. (1995). Microgenetic study of change: What has it told us? *Psychological Science, 6*(3), 133–139. doi:10.1111/j.1467-9280.1995.tb00322.x

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J. ... Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32–37. doi:10.1145/1929887.1929902

Lewis, M. D. (2000). The promise of dynamic systems approaches for an integrated account of human development. *Child Development, 71*(1), 36–43.

Liao, Y.-K. C., & Bright, G. W. (1991). Effects of computer programming on cognitive outcomes: A meta-analysis. *Journal of Educational Computing Research, 7*(3), 251–268. doi:10.2190/E53G-HH8K-AJRR-K69M

Lightfoot, C., Cole, M., & Cole, S. (Eds.) (2009). *The development of children* (6th ed.). New York: Worth.

Logo Foundation (2000). *What is Logo?* Retrieved from http://el.media.mit.edu/logo-foundation/logo/index.html

MA DOE (2006). *Massachusetts science and technology/engineering curriculum framework.* Retrieved from http://www.doe.mass.edu/frameworks/scitech/1006.pdf

MA DOE (2008). *Massachusetts technology literacy standards and expectations.* Retrieved from http://www.doe.mass.edu/odl/standards/itstand.pdf

Marshall, P. (2007). Do tangibles enhance learning? In *Proceedings of the First International Conference on Tangible and Embedded Interaction* (pp. 163–170). New York: ACM. doi:10.1145/1226969.1227004

Martin, F., Mikhak, B., Resnick, M., Silverman, B., & Berg, R. (2000). To mindstorms and beyond: Evolution of a construction kit for magical machines. In A. Druin & J. A. Hendler (Eds.), *Robots for kids: Exploring new technologies for learning* (pp. 9–33). San Francisco: Morgan Kaufman.

McDevitt, T. M., & Ormrod, J. E. (2002). *Child development and education.* Upper Saddle River, NJ: Merrill/Prentice Hall.

McGill, T. J., & Volet, S. E. (1997). A conceptual framework for analyzing students' knowledge of programming. *Journal of Research on Computing in Education, 29*(3), 276–297.

NAEYC & Fred Rogers Center (2012). *Technology and interactive media as tools in early childhood programs serving children from birth through age 8.* Retrieved from http://www.naeyc.org/files/naeyc/file/positions/PS_technology_WEB2.pdf

New Media Consortium (2005). *A global imperative: The report of the 21st century literacy summit.* Retrieved from http://www.nmc.org/pdf/Global_Imperative.pdf

Nir-Gal, O., & Klein, P. S. (2004). Computers for cognitive development in early childhood: The teacher's role in the computer learning environment. *Information Technology in Childhood Education Annual, 2004*(1), 97–119.

Papert, S. (1993). *Mindstorms: Children, computers, and powerful ideas* (2nd ed.). New York: Basic Books.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology, 2*(2), 137–168. doi:10.1016/0732-118X(84)90018-7

Pea, R. D., Kurland, D. M., & Hawkins, J. (1985). Logo and the development of thinking skills. In M. Chen & W. Paisley (Eds.), *Children and microcomputers: Research on the newest medium* (pp. 193–317). Beverly Hills, CA: Sage.

Peppler, K. A., & Kafai, Y. B. (2007). From SuperGoo to Scratch: Exploring creative media production in informal learning. *Learning, Media and Technology, Special Issue: Media Education Goes Digital, 32*(2), 149–166. doi:10.1080/17439880701343337

Perlman, R. (1976). *Using computer technology to provide a creative learning environment for preschool children* (AI Memo 360: Logo Memo No. 24). Cambridge, MA: MIT Artificial Intelligence Laboratory.

Resnick, M. (2006). Computer as paintbrush: Technology, play, and the creative society. In D. Singer, R. Golikoff, & K. Hirsh-Pasek (Eds.), *Play = learning: How play motivates and enhances children's cognitive and social-emotional growth.* Oxford, UK: Oxford University Press.

Resnick, M. (2007). Sowing the seeds of a more creative society. *Learning & Leading with Technology, 35*(4), 18–22. doi:10.1108/14777280710828549

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. doi:10.1145/1592761.1592779

Revelle, G. (2013). Applying developmental theory and research to the creation of educational games. *New Directions for Child and Adolescent Development, 2013*(139), 31–40. doi:10.1002/cad.20029

Salomon, G., & Perkins, D. N. (1987). Transfer of cognitive skills from programming: When and how? *Journal of Educational Computing Research, 3*(2), 149–169. doi:10.2190/6F4Q-7861-QWA5-8PL1

Siegler, R. S., & Crowley, K. (1991). The microgenetic method: A direct means for studying cognitive development. *American Psychologist, 46*(6), 606–620.

van Geert, P. (1998). A dynamic systems model of basic developmental mechanisms: Piaget, Vygotsky, and beyond. *Psychological Review, 105*(4), 634–677. doi:10.1037//0033-295X.105.4.634-677

Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society—Series A, 366, 3717–3725.* doi:10.1098/rsta.2008.0118

Manuscript received October 12, 2012 | Initial decision March 26, 2013 | Revised manuscript accepted May 20, 2013