Tech Know Learn (2017) 22:443–463
DOI 10.1007/s10758-017-9328-x

ORIGINAL RESEARCH

Introducing Computational Thinking to Young Learners: Practicing Computational Perspectives Through Embodiment in Mathematics Education

Woonhee Sung[1] · **Junghyun Ahn**[1] · **John B. Black**[2]

Published online: 28 July 2017
© Springer Science+Business Media B.V. 2017

Abstract A science, technology, engineering, and mathematics-influenced classroom requires learning activities that provide hands-on experiences with technological tools to encourage problem-solving skills (Brophy et al. in J Eng Educ 97(3):369–387, 2008; Matarić et al. in AAAI spring symposium on robots and robot venues: resources for AI education, pp 99–102, 2007). The study aimed to bring computational thinking, an applicable skill set in computer science, into existing mathematics and programming education in elementary classrooms. An essential component of computational thinking is the ability to think like a computer scientist when confronted with a problem (Grover and Pea in Educ Res 42(1):38–43. doi:10.3102/0013189X12463051, 2013). Computational perspectives (Berland and Wilensky in J Sci Educ Technol 24(5):628–647. doi:10.1007/s10956-015-9552-x, 2015) refer to the frame of reference programmers or computer scientists adopt when approaching a problem. The study examined the effects of taking computational perspectives through various degrees of embodied activities (i.e., full vs. low) on students' achievement in mathematics and programming. The study employed a 2 (full vs. low embodiment) × 2 (with vs. without computational perspective taking) factorial condition to evaluate four learning conditions from a combination of embodiment and computational perspective-taking practice. The results from this experimental study ($N = 66$ kindergarten and first graders) suggest that full-embody activities combined with the practice of computational perspective-taking in solving mathematics problem improved

✉ Woonhee Sung
Ws2345@tc.columbia.edu

Junghyun Ahn
Ja2178@tc.columbia.edu

John B. Black
Black@tc.columbia.edu

[1] Department of Mathematics, Science and Technology, Teachers College, Columbia University, 525 W. 120th Street, New York, NY 10027, USA

[2] Department of Human Development, Teachers College, Columbia University, 525 W. 120th Street, New York, NY 10027, USA

mathematics understanding and programming skills as demonstrated in *Scrath Jr.* among novice young learners. Moreover, the practice of using a computational perspective significantly improved students' understanding of core programming concepts regardless of the level of embodiment. The article includes recommendations for how to make the computational thinking process more concrete and relevant within the context of a standard curriculum, particularly mathematics.

Keywords Computational thinking · Embodied cognition · Elementary education · Programming · Mathematics · Computational perspectives · STEM

1 Introduction

Recent technological advancements in science, technology, engineering, and mathematics (STEM) learning have raised questions among researchers and educators about how to support children's STEM learning using these new technologies. Since STEM concepts are shared across science, mathematics, and engineering, attention should be paid to fundamental problem-solving skills that are applicable to STEM domains.

Wing's (2006) influential article, which introduced the concept of computational thinking, ignited passionate debate about the aims and priorities of STEM education in K-12 contexts. The notion that computational thinking is a fundamental thinking skill in K-12 education is not new to the 21st century. In the 1970s, Papert (1972) described computation as the fundamental ingredient of educational innovation and went on to create LOGO programming, which turns abstract and distant computation concepts into concrete instruments. Consequently, programming education, an essential way to teach and demonstrate computational thinking, gained traction in K-12 education in the United States.

Since then, many scholars have discussed the cognitive benefits and outcomes from learning programming and its relation to other subject areas such as mathematics (Feurzeig et al. 2011; Kurland and Pea 1985; Papert 1972, 1980). While there are different views on the mental activities engaged by programming and their expected educational benefits (Pea and Kurland 1984), Wing's (2006) view is in line with that of Pea and Kurland (1984), who stated that "through learning to program, children are learning much more than programming, far more than programming facts … children will acquire powerfully general higher cognitive skills such as planning abilities, problem-solving process itself" (p. 138). This belief is consistent with computational thinking in the 21st century, defined as "thinking involving solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (Wing 2006, p. 33).

With the rapid rise of programming instruction in K-12 education, students are increasingly exposed to programming tools at an early age, with high expectations for game design or programming outcomes in educational settings. To date, studies of students' learning of computational concepts and thinking skills have employed programming languages such as Scratch (Burke 2012; Resnick et al. 2009), Hopscotch (Fryer 2014), and tangible programming with the use of physical blocks (Kazakoff and Bers 2012) in K-6 settings, focusing on how to improve programming skills through the use of diverse tools rather than on applying programming concepts or computational thinking in other STEM subjects.

However, in elementary education, the fluency of using game and programming software or the ability to design games is limited, and the use of tools is regulated in formal education. Thus, finding ways to foster computational thinking and to incorporate

computer programming into existing K-12 curricula is considered to be critical in recent years. Along the same lines, Lye and Koh (2014) pointed out the lack of intervention approaches that study factors encouraging students' use of computational thinking and the transfer effects on other skills that can be applied across nonprogramming domains. Pea and Kurland (1984) expressed this concern, emphasizing the importance of organizing learning experiences to lead students to new ideas and opportunities to build their own understanding of computational concepts rather than focusing on what skills to teach. Berland and Wilensky (2015) coined the term *computational perspectives* to refelect the approach computer programmers or scientists take to solving a problem. Berland and Wilensky (2015) pointed out that computational thinking focuses on the use of technical skills to do programming whereas computational perspectives enable learners to apply "computation across domains that are not necessarily computer scientific" (p. 3).

If learning activities are carefully designed to embed computational perspectives, students can engage computational thinking skills in other domains that are not computer scientific. Thus, the purpose of the present study was to identify factors in designing educational activities appropriate for elementary classrooms that can integrate computational thinking with non-computing domains. The study adopted an embodied approach rooted in the premise that bodily activities interacting in a perceived physical world support the learning of abstract concepts (Barsalou 2008; Glenberg 2008, 2010; Wilson 2002). Alimisis (2013) introduced embodied approach as an innovative way to introduce STEM-related activities for younger children.

The authors investigated the effects of two factors, embodied activity and computational perspective-taking practice integrated in mathematics learning: (1) full-embody (whole body movement) versus low-embody (hand gestures) and (2) computational perspective-taking practice versus without taking computational perspectives. These interventions rely on students' physical and cognitive activities that are plugged into mathematics learning contexts, not on technological tools, to address the current shortage of resources in STEM topics among underserved groups (Matarić et al. 2007). Thus, the study sought to identify effective factors for designing hands-on learning activities that enhance mathematics and computational thinking skills for underrepresented minority students. By adopting embodied cognition and a computational perspective in the learning activities, the study aimed at proposing a curriculum that lowers dependence on technological tools while maximizing students' participation in their learning process.

The proposed curriculum is expected to inform educators and researchers on how to design and implement grade-appropriate programming in the K-12 education. Specifically, based on the findings, we offer suggestions on how to make the computational thinking process more concrete and relevant within the context of a standard curriculum, particularly mathematics, so students can apply this thinking skill in STEM fields such as programming and robotics.

2 Theoretical and Empirical Background

2.1 Embodied Approach

To truly engage a child's interest in STEM education, experiential, hands-on education has long been recognized as superior for learning new material by conveying real-world meaning to abstract knowledge (Matarić et al. 2007). Based on the premise that cognitive

processes are deeply related to bodily activities in the physical world, multiple research areas support the claim that embodiment is a powerful learning approach in learning abstract domains (Barsalou 2008; Barsalou et al. 2003). In particular, the connection between knowledge representation and the bodily states are emphasized in embodied cognition (Barsalou 2008; Glenberg 2008, 2010; Wilson 2002; Wilson and Golonka 2013). Based on the belief that kinesthetic interactions grounded in concepts lead to conscious thinking and more concrete experiences, embodied cognition has been discussed as an innovative way to introduce abstract concepts such as mathematics, robotics, science concepts, or programming for children (Alimisis 2013). Particularly in elementary STEM classrooms, the focus is on learner-centered and constructionist approaches to design motivating and engaging activities. Embodied cognition recognizes the importance of learners participating with the physical world through active interaction to construct conceptual metaphors emerging from concrete sensorimotor experiences (Abrahamson and Howison 2010; Abrahamson and Trninic 2015; Bamberger and DiSessa 2003).

Importantly, research studies share the assumption that cognition is grounded in the sensorimotor activity of our bodies; thus, expanding the body's physical interaction within conceptually grounded environment provides "fertile soil onto which we can lay the seeds of new learning" (Lindgren 2014, p. 40). Therefore, an embodied approach is a valuable resource to learn abstract and symbolic concepts or literacies that are unfamiliar to young learners, such as programming languages, mathematics, and technology. For example, Fadjo (2012) evaluated different types of embodiment in programming contexts. Fadjo (2012) asked students to physically enact predefined scenarios and coding sequences, which he referred to as a form of direct embodiment. Learners embody themselves as agents executing coding commands, taking the perspective of a robot or virtual character following coding. Surrogate embodiment is also a type of embodiment; however, here learners manipulate an external surrogate, not their own bodies. In Fadjo's (2012) study, direct embodiment had a significant effect on developing programming skills, especially conditional sequences whereas surrogate embodiment provided a unique opportunity for the instruction of arithmetic topics during video game design (Fadjo et al. 2009a, b). Thus, in our study, the dynamic environment allowing physical interactions grounded in the target concept became the key factor and employed direct embodiment (learners engaged their own bodies) and surrogate embodiment (learners manipulated an external surrogate without engaging their own bodily movement).

2.2 Level of Embodiment

Several research studies have found evidence of the relationship between the embodied approach and STEM content learning. For example, in a study about solving gear problems, Schwartz and Black (1996) argued that spontaneous hand gestures helped learners imagine the correct movement of the gears by physically instantiating mental models. This line of research has continued with the development of digital devices and touchable screens. Before incorporating the haptic channel feedback, Chan and Black (2006) found that students who were able to directly manipulate the animation using multimodal representation actively participated in forming mental models. The haptic channel using a 2-D force feedback stick indicated that students experiencing 2-D force feedback through a joystick outperformed students without 2-D force feedback (Hallman et al. 2009).

Further, Huang et al. (2011) achieved consistent results when using a 3-D force feedback joystick that provided a full range of movement, which a 2-D joystick is not capable. That is, when learning the law of conservation of energy, the use of a 3-D force feedback

provided an embodied experience that led to better construction of mental models and learning outcomes of abstract physics concept among younger learners. These findings suggest that a higher level of feedback engagement supports perceptual experience among learners by providing concrete simulations that help them understand abstract physics concepts. When learners' bodily movement was deeply grounded in simulations, their performance on an immediate post-test and near transfer test improved (Huang et al. 2011), showing that their mental model was strengthened and became more flexible at adapting as the level of control over the simulation and the movement involved expanded.

Recent studies (e.g., Johnson-Glenberg et al. 2014) support this claim by showing that learning activities involving a higher level of embodiment lead to a greater chance of retrieval and retention. Johnson-Glenberg et al. (2014) compared the learning gains of students who were physically swinging an object versus students clicking the mouse on a computer simulation about centripetal force. In the immediate post-test, all students demonstrated learning gains; however, significant learning gain in physics knowledge was reported only on the follow-up test of students from the high-embodied learning condition. The authors concluded that the students in the latter condition showed better retrieval than the low-embodied group, who demonstrated decreased knowledge.

When learning contents with rich visuospatial properties, such as counting, numerical magnitude, or number line, the high-embodied condition provides benefits from acting on a visual-kinesthetic interface (Abrahamson and Howison 2010; Bamberger and DiSessa 2003). Thus, the high-embodied condition in this study involved full body movement with a larger-size number line whereas learners in the low-embodied condition used their hands with a number line drawn on a piece of paper the same size as the testing material.

3 Computational Thinking and Computational Perspective

Computational thinking is mentioned as an essential skill for the twenty first century, giving learners a different framework for visualizing and analyzing problems (Einhorn 2011). Wing (2006) described computational thinking as a set of thinking skills, habits, and approaches that are integral to solving problems and designing systems from a scientist's perspective. Her vision provided a springboard to a more comprehensive approach to designing and developing computational thinking in K-12 education. The effort to bring computational thinking in childhood education began back in the 1980′s by Papert (1980), although it was called programming education rather than computational thinking. Papert (1980) advocated for programming education through LOGO programming, which provided a constructive programming environment to improve children's procedural thinking and mathematics understanding (Clements and Battista 1989). After the advent of visual programming tools (i.e., Scratch, Scratch Jr., Hopscotch), robotics (i.e. LEGO Wedo or Mindstorm), and tangible programming bricks, attention has been paid to the use and function of tools for computational thinking development in K-12 education (Lye and Koh 2014). For example, computer programming tools developmentally appropriate for children were found to be beneficial for young learners' cognitive, problem-solving, and sequencing skills (Bers 2008, 2010; Clements and Sarama 2002; Kazakoff and Bers 2012; Lee et al. 2011). In addition, programming experiences with tangible robots or virtual sprites have been found to improve the development of domain knowledge as well as skills described as computational thinking. Little attention has been paid to the design of learning procedures that computational thinking can be exhibited without technology tools.

Moreover, computational thinking is not only related to the programming domain but also to problem-solving practices requiring a specific mode of thinking that computer scientists or programmers demonstrate.

The National Research Council (NRC) highlighted mathematics and computational thinking as essential practices for K-12 science and engineering education (NRC 2012). Furthermore, the International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) developed an operational definition of computational thinking as algorithmic thinking to formulate problems with a tool and organizing and representing data through abstraction with the use of simulation (ISTE 2011). Although there is no widely agreed-upon definition of computational thinking in K-12 settings, various interpretations suggest that computational thinking can be infused in other subject domains by designing learning tasks that engage core mental processes of computational thinking.

In response to this necessity, this study explored what type of instructional activities can support thinking-doing, not just doing. Therefore, the authors designed interventions that implemented the computational mode of thinking in the context of mathematics (number line estimation, numeracy, and arithmetics) to guide students' experience. This mode of thinking is based upon Berland and Wilensky's (2015) definition of computational perspectives, which emphasizes the way computer programmers think when approaching a problem, through a logical, analytical, and constructive lens.

Programming is a tool that alters learners' viewpoint as a user to a programmer who answers the question of "how would I make the machine to solve this problem?" by providing commands to the machine recursively (Wing 2008). This "necessarily explicit nature of programming" (Papert 1980; Pea and Kurland 1984, p. 142) enables students to practice a computational perspective. However, for novice learners, especially younger elementary students, learning to use programming language may require developmental reorganization in understanding programming constructs, tool functions, and, more importantly, the computational perspective. Concerns about programming education among novice learners were discussed by Pea and Kurland (1984), who asked, "How can we organize learning experiences so that in the course of learning to program students are confronted with new ideas and have opportunities to build them into their own understanding of the computer system and computational concepts? What component mental processes are engaged in programming activities?" (p. 140).

The current study was guided by these questions to design effective learning activities that incorporate the computational perspective in non-programming domains as a means of equipping students with problem-solving skills that can be transferred to programming tasks. Thus, the interventions were designed to practice computational thinking without the programming application, but in the mathematics domain. Participants experienced the computational perspective in the general problem-solving task and were provided with a programming task that required the application of computational concepts acquired from the activities.

3.1 Computational Thinking in Mathematics

With the goal to foster an application of this mode of thinking, we chose mathematics as the content area to exercise the computational perspective. Mathematical ability is often viewed as a core factor in predicting students' ability to learn computer programming and, thus, is required for that area of study (Pea and Kurland 1984). Mathematical thinking is closely related to computational thinking because solving a mathematical problem is a

process of construction (Feurzeig et al. 2011) that requires an analytic problem-solving perspective, which is unique and fundamental to computer programmers or scientists (Berland and Wilensky 2015). Studies using LOGO programming in the 1980′s indicated that programming activities with young students facilitated learning the relationship between numerical magnitude and length estimation of a number line (Clements and Battista 1989; Hughes and Macleod 1986; Robinson and Uhlig 1988). Thus, the learning activities of this study added "concrete" and "analytical" instances of reasoning that are inherent in computational thinking to mathematics problem solving, particularly number sense and number line estimation.

4 Embodied Activity Incorporating Practice From a Computational Perspective

Many cognitive scientists have studied how bodily action affects conceptual development and, in turn, proposed models explaining how learning abstract concepts benefit from concrete embodied experiences. The design of the current study adopted an embodied approach and computational perspectives for learning mathematical concepts and engaging computational thinking in a programming context targeting young learners.

Research studies verify that learning activities involving a high level of embodiment lead to a higher chance of retrieval and retention than using low-embody activities such as hand movements (Johnson-Glenberg et al. 2014). These findings suggest that a greater degree of bodily engagement supports the perceptual experiences of learners by providing concrete experiences.

In this study, the authors investigated different degrees of embodiment (full vs. low) as the first factor influencing mathematics and programming performance. In terms of how to design learning activities that incorporate the computational perspective into a mathematics problem-solving activity, the design was based on the conceptual framework of embodiment in formal educational settings, called instructional embodiment within STEM content (Black et al. 2012; Fadjo et al. 2009a, b).

The current study employed two forms of instructional embodiment from the conceptual framework introduced by Fadjo (2012), direct embodiment (DE) and surrogate embodiment (SE). DE is the physical enactment of predefined scenarios or sequences that contain explicit and implicit cues for movement (Fadjo 2012). This was applied in role-play activities that engaged the students' bodily movement without involving a surrogate but consisted of moving their bodies to execute the problem-solving steps. SE, in turn, is a type of physical enactment where the learner controls and manipulates the movement of an external surrogate, such as a virtual character or physical object (human) (Fadjo 2012). The authors found conceptual similarity between surrogate embodiment and the computational perspective in the sense that manipulating a surrogate resembles the thought processes of a programmer. Operating a surrogate requires students to display explicit knowledge in the form of articulation to provide commands to manipulate the surrogate. SE requires students to imagine the expected performance of a surrogate and transform the movement description into a form of commands to manipulate the surrogate, which is a peer learner in this study. Therefore, students in the SE condition took the perspective of programmers, which is defined as a core concept of a computational perspective. Finally, the study divided participants into groups by full and low level of embodiment in combination with

the use of computational perspectives, either employing a surrogate or not. Next, the research design describes how the groups were defined.

5 Method

5.1 Participants

We examined the effects of embodied activity and computational perspective practice as an instructional strategy in children's development of early mathematics and programming skills through experiencing computational perspectives in embodied activities. Participants were 66 kindergarten and first-grade students (36 males, 30 females) enrolled in an after-school coding program in two neighboring New York City public schools. Both ethnically diverse schools consist of 25–50% Hispanic, 39–47% Black, 20% or less White, and less than 6% Asian students. The percentage of economically disadvantaged students in the two schools is 47 and 88%, respectively. The majority of study participants are Hispanic, Black, or Asian, consistent with the prevailing ethnicity of their schools.

The students performed different types of embodied activities before programming on the iPad-based Scratch Jr. The first task consisted of filling in a number line with equal interval lengths on a floor grid (full-embody group) or on a paper (low-embody group). Participants in the Computational Perspective Practice (CPP) group performed the role of programmer, which involved (a) commanding a surrogate and (b) providing the procedural steps for solving a problem in combination with bodily and hand movements. The participants without CPP used bodily or hand movements to solve the problem without communicating with peer surrogates.

5.2 Research Design

The purpose of the study was to examine the impact of embodied interaction with or without CPP in children's conceptual understanding in mathematics and programming. The research question aims to evaluate the main effects of the degree of embodiment (full vs. low) and the presence of CPP (surrogate role present vs. no role assigned) on students' mathematics and programming learning. Thus, the study used intervention approaches designed as a 2 × 2 factorial experiment (see Table 1) with two independent variables, resulting in four experimental groups. Testing the effects of the four conditions from the combination of the two independent variables helped uncover evidence on how different types of learning conditions lead to different levels of understanding in mathematics and programming.

Table 1 2 × 2 Factorial experimental groups

Computational perspective practice (CPP)	Degree of embodiment	
	Full embodiment (Full body)	Low embodiment (Low degree)
With CPP (Surrogate/commander role-play)	Full with CPP	Low with CPP
Without CPP (no roles assigned)	Full without CPP	Low without CPP (Control group)

The researchers randomly assigned three after-school coding classes into the four conditions.

1. Full-embody activity with CPP through surrogate/commander role-play (full embodied & surrogate/commander role-play)
2. Full-embody activity without CPP (full embodied & no commander role)
3. Low degree of embodied activity with CPP through surrogate/commander role-play (low embodied & surrogate/commander role-play)
4. Low degree of embodied activity without CPP (control group)

Through embodied activity with CPP, students practiced (a) commanding a surrogate and (b) providing the procedural steps to solve a problem in the domain of mathematics. The level of bodily engagement was combined with the presence of CPP. Direct embodiment with a high level of bodily movement activity (full without CPP) was implemented in the form of role-playing, where students directly acted on the physical numerical representation (a number line) to demonstrate the given tasks. Embodying computational perspectives with a high level of bodily engagement (full with CPP) also used physical representation, but students were asked to manipulate an external surrogate, another student, to perform the given tasks.

The manipulation of a surrogate under embodiment within the CPP condition resembles the concept of commanding as students manipulate sprites in Scratch Jr., which is intended to be a unique characteristic of computational thinking skills as defined above. In order to operate surrogates to complete a task, students need to provide the correct steps by decomposing the given problem and then verbalizing the procedures required to solve the problem. This series of thought processes was used with the presence of CPP. CPP required students to verbalize unconscious movement, thereby making it conscious. This conscious effort was maintained in the CPP condition with a low level of embodiment (low with CPP), where students used hand drawings instead of bodily movement. A low level of enactment without CPP (low without CPP) excluded both the effect of bodily engagement and a computational perspective; thus, it served as the control condition.

5.3 Procedure

This experimental study was conducted over five sessions, 1 h per session, in two New York City public elementary schools as part of their after-school curriculum.

5.3.1 Session 1 (Pre-test)

In the first week, a pre-test was administered. Participants completed a paper-based pretest, measuring their prior knowledge with number line, counting, number ordering, addition, subtraction, and magnitude comparison. Next, Scratch Jr. was introduced to allow students to explore this unfamiliar software. No specific instruction on coding was provided, but students were encouraged to figure out how the software worked.

5.3.2 Session 2 (Embodiment Intervention 1)

In week 2, three classes were divided into the four conditions, and the number line creation activity began. The full-embody (with/without CPP) group created a number line on a floor mat whereas the low-embody (with/without CPP) group created one with paper and pencil. Students spent 30 min to create a line by placing the numbers zero to 10 using a

"measurement arrow stick," which resembled the coding block used in Scratch Jr. The full-embody group used a larger measurement block card whereas the low-embody group used a smaller size suitable for the size of paper used. Then, across all four groups, the students played with Scratch Jr. to complete a number line using programming blocks, as shown in Fig. 1.

5.3.3 Session 3 (Embodiment Intervention 2)

After the first intervention, the four groups moved to the second phase of the intervention. The second embodied activity also used a blank number line on the floor (for full-embody group) and with paper and pencil (for low-embody group). Students were asked to perform or draw how a simple equation (e.g., $2 + 5 = ?$) could be solved using a given blank number line and a measurement stick. Using the "measurement arrow stick" (full-embody group) or "measurement arrow card" (low-embody group), students were asked to either physically move on the number line or draw on the paper-based number line to solve the equation. After spending 30 min on this intervention, students completed a paper-based post-test using the same format and content as the pre-test. A 15-min time limit was set for completing the post-test for all four groups.

5.3.4 Session 4 (Scratch Jr. Programming)

After completing 2 weeks of embodied activities using number line estimation, addition, and subtraction skills, students were asked to perform a transfer task using programming. This novel task on the programming software Scratch Jr. required students to program two sprites, a cat and a basketball, so that the cat would throw a ball into the basket (see Fig. 2). Limited guidance was provided, and the same level of instruction was given to each group. An hour was allowed to complete the task.

Fig. 1 First programming project of filling in a number line using programming blocks on Scratch Jr.

Fig. 2 Second programming project of applying the concept of addition using two sprites in Scratch Jr.

5.3.5 Session 5 (Delayed Test)

Upon returning to the program after 1 week of vacation, students completed a delayed test on number line estimation and number sense.

5.4 Data Sources

The materials used for the study consisted of paper-based pre-, post-, and delayed tests to measure participants' mathematics learning outcomes from unplugged activities in four different conditions. Three dependent measures were administered to evaluate participants' number line sense, addition, and subtraction skills, and two dependent measures were observed from programming artifacts.

5.4.1 Pre- and Post-tests

The mathematics test included number line estimation, counting, addition, subtraction, and numerical magnitude comparison. All items were developed by the researchers and adapted from the New York State P-12 Common Core Learning Standards.[1] A post-test, which was identical to the pre-test, was administered after the embodied activity intervention to measure improvements within and between groups. These paper-based tests asked students to fill in the blank linear number line from 1 to 10 and answer questions related to counting on number lines and addition skills. Number line estimation accuracy was graded based on the Percent of Absolute Error (PAE), which calculates the discrepancy between the estimated position and the target number presented, divided by the total numerical range (Booth and Siegler 2008). The post-test included one additional item of creating an unfamiliar number line from 0 to 8. Students at this age are mostly exposed to the 10th unit and are, therefore, familiar with a number line ending with the 10th unit, such as 10, 20, or 100. Since the study intended to investigate how students' interval estimation

[1] https://www.engageny.org/resource/new-york-state-p-12-common-core-learning-standards.

improves based on the linear measurement on a number line activity and procedural/sequential thinking skills practices, the length of an unfamiliar number line was also tested. For the additional number line estimation task, the slope value representing the linearity other estimated number line was calculated and compared across groups.

5.4.2 Delayed Test

The delayed test consisted of two number line estimation tasks, addition, subtraction, a position to number task (PN task), and a number to position task (NP task). The PN task (Siegler and Opfer 2003) asks students to estimate the number that corresponds to the position marked on a number line and the NP task shows a number for students to estimate its position on a line.

5.4.3 Programming Skills

After two intervention sessions of embodied activities, participants completed programming challenges by creating visual coding blocks in Scratch Jr. These programming products were evaluated in the form of success scores and programming efficiency scores. Considering the possible number of errors participants can make, a success score was recorded for each sprite. For programming Cat, 0 (more than 3 errors), 1 (fewer than 2 errors), and 2 (complete success) was given. Programming Basketball was graded as 0 (more than 4 errors), 1 (2–3 errors), 2 (1 error), and 3 (complete success). Figure 3 shows an example of efficient coding produced by students. When participants created more sophisticated and efficient coding blocks by using addition, pattern recognition, and fluent coding skills, one point was given for each strategy use.

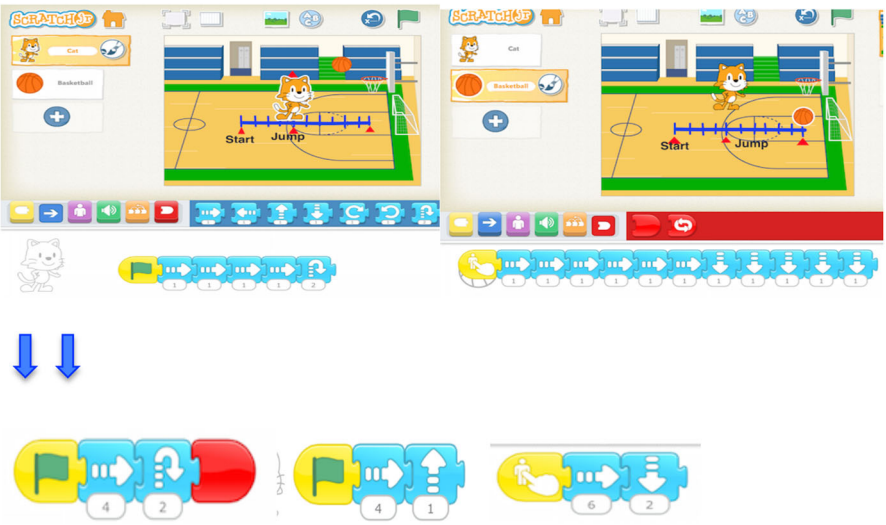

Fig. 3 Example programming blocks of efficient coding

6 Results

This study examined the impact of two factors, the degree of embodiment and the presence of the computational perspective practice, and their combined impact. Learning outcomes were measured in two domains: mathematics (number line sense and arithmetic skills) and programming (correctness and efficiency). Students' prior knowledge of mathematics showed no difference among the groups $F(3, 45) = .704$, $p = .555$. Students had neither prior experiences with the programming tool nor knowledge about programming concepts.

6.1 Posttest

A 2×2 ANOVA was conducted on the post-test. The degree of embodiment, $F(1,55) = 8.632$, $p = .005$, partial $\eta^2 = .16$, had a statistically significant impact on the post-mathematics test. The full-embody group ($n = 35$, $M = 8.56$, $SD = 0.57$) scored higher than the low-embody group ($n = 24$, $M = 5.90$, $SD = 0.70$). A one-way ANOVA revealed that the difference in the post-test score between the control group, low-embody without CPP ($n = 14$, $M = 5.00$, $SD = 4.47$), and the full-embody with CPP group ($n = 19$, $M = 8.63$, $SD = 2.79$) was statistically significant, $F(3,55) = 3.84$, $p = .01$ (Table 2).

Total post-test scores showed significant differences between the high and low level of embodiment; however, the number line estimation task (0–10) did not show differences across groups (Fig. 4).

Another applied item estimation task was tested, in which students estimated the position of 0 to 8 transformed into a slope value representing the linearity of an estimated number line. The estimated position of 0 to 8 (y) should increase linearly with the actual position (x) of 0 to 8 with a slope of 1.00. To examine how close the slope produced from students in each group was to the perfect slope of 1.00, the deviation value between students' estimated number line slopes and a perfect value of 1.00 was calculated and named Slope Error. A nonparametric Mann–Whitney test was performed because Levene's test failed to support the homogeneity of variance of the frequency data. The test indicated that the slope error value was greater for the low-embody group ($Mdn = 0.36$) than for the full-embody group ($Mdn = 0.13$), $U = 98.50$, $p = .000$ (Fig. 5).

Overall, our posttest analysis found that the full-embody group performed better than the low-embody group. Moreover, the full-embody with CPP group achieved higher than the low-embody without CPP group. The item estimating 0 to 8 on a line showed a lower

Table 2 Two-way analysis of variance on the posttest by degree of embodiment and CPP factor

Source	SS	df	MS	F	Sig.	Partial η^2
Degree of embodiment	99.196	1	99.196	8.632	.005**	.136
CPP	13.020	1	13.020	1.133	.292	.020
Degree of embodiment × CPP	09.714	1	09.714	.845	.362	.015
Error	632.021	55	11.491			
Total	4114.00	59				

$R^2 = .173$, adj. $R^2 = .128$

Design: Intercept + Degree of embodiment + CPP + Degree of embodiment × CPP

* $p < .05$, ** $p < .01$

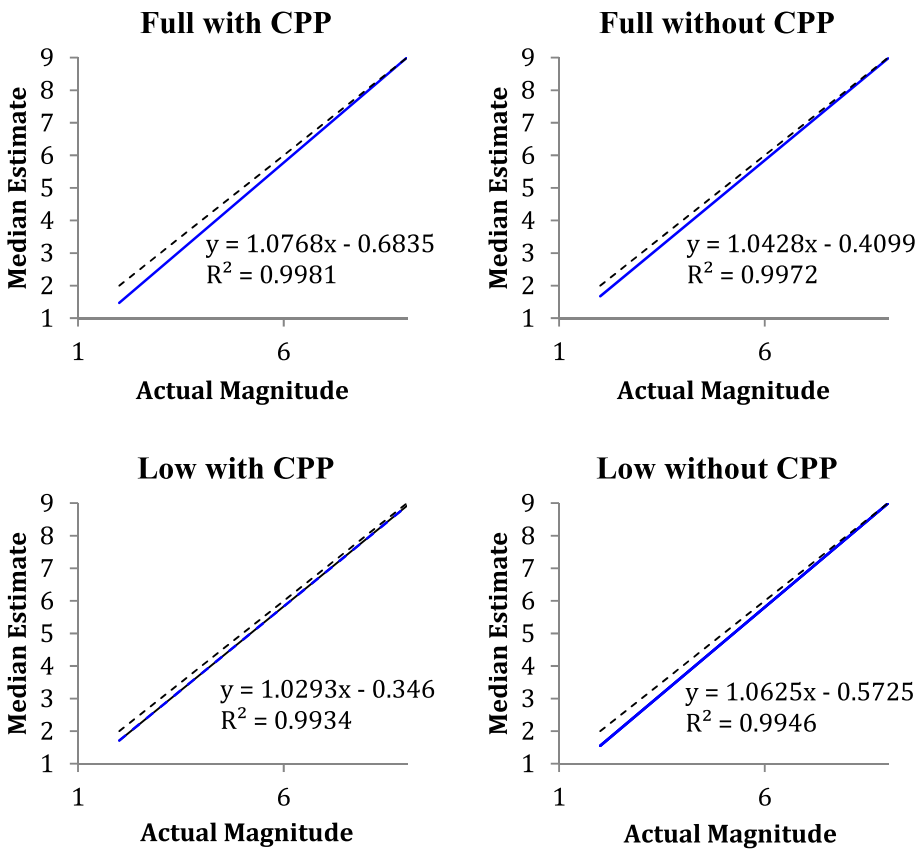

Fig. 4 Post-test linearity of number line estimation: group median data. Slope of each group's linearity is defined as the slope of the linear (*solid*) line. PAE is defined as the distance from the *dashed, diagonal line* (perfect linearity 1.0). The variance accounted for by the linear line of each group (R^2) is .998 for full with CPP, .997 for full without CPP, .993 for low with CPP and .994 for low without CPP

Slope Error in the full-embody group whereas the low-embody group demonstrated a larger error size.

6.2 Delayed Test

The delayed test examined number line estimation (0 to 12 and 0 to 6), arithmetic skills, and position to number (PN task). In terms of the accuracy of the number line estimation, the difference between each participant's number line slope and a perfect slope of 1.00 was calculated. A Mann–Whitney test comparing the error value indicated that the group with CPP ($Mdn = 0.27$) had a significantly smaller error size than the group without CPP ($Mdn = 0.37$), $U = 185.00$, $p = .021$. Considering the result of the post-test number line estimation task and the applied item, the significance of the CPP factor in the delayed test was remarkable.

For the PN task, the difference between the estimated number and the correct number was calculated and converted to a score, with a higher value indicating a smaller error size. A 2 × 2 ANOVA analysis indicated that both the degree of the embodiment factor,

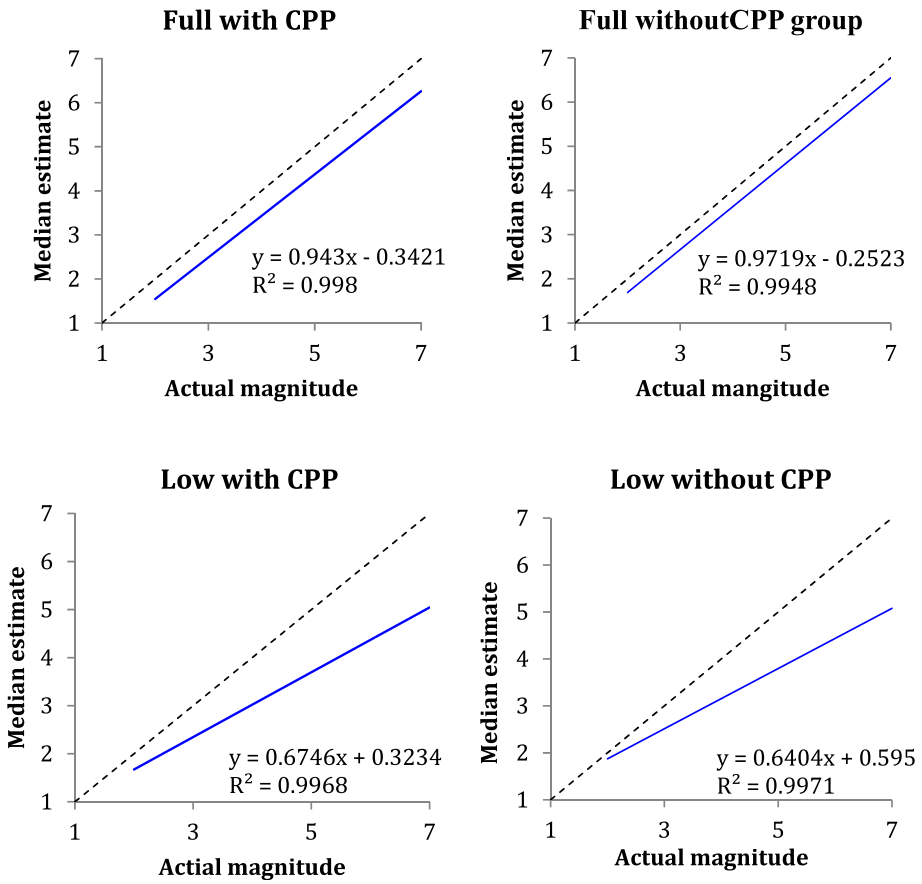

Fig. 5 Applied number line item linearity: group median data

$F(1,43) = 8.258$, $p = .006$, partial $\eta^2 = .16$, and the presence of CPP, $F(1,43) = 4.413$, $p = .04$, partial $\eta^2 = .09$, had a statistically significant impact on PN task performance. The full-embody group ($n = 26$, $M = 4.69$, $SD = 1.69$) had a higher score than the low-embody group ($n = 21$, $M = 2.95$, $SD = 2.03$) and the group with CPP ($n = 22$, $M = 4.63$, $SD = 2.01$) scored higher than the group without CPP ($n = 25$, $M = 3.28$, $SD = 1.86$) (Table 3).

A one-way ANOVA post hoc test showed results that were consistent with the post-test; that is, the full-embody with CPP group ($n = 14$, $M = 5.14$, $SD = 1.83$) significantly outperformed the low-embody without CPP group ($n = 13$, $M = 2.46$, $SD = 1.89$), with $F(3,43) = 5.06$, $p = .004$. A 2×2 ANOVA analysis on the delayed test total score revealed significant results consistent with the PN task analysis for both factors, degree of embodiment $F(1,43) = 7.959$, $p = .007$, partial $\eta^2 = .16$, and the presence of CPP, $F(1,43) = 4.331$, $p = .04$, partial $\eta^2 = .09$. The full-embody group ($n = 26$, $M = 6.69$, $SD = 2.34$) and the group with CPP ($n = 25$, $M = 6.64$, $SD = 2.74$) demonstrated better understanding and knowledge retention than the low-embody group ($n = 21$, $M = 4.38$, $SD = 2.71$) and the without CPP group ($n = 25$, $M = 4.80$, $SD = 2.50$).

Table 3 Two-way analysis of variance on the PN task by degree of embodiment and CPP factor

Source	SS	Df	MS	F	Sig.	Partial η^2
Degree of Embodiment	26.908	1	26.908	8.258	.006**	.161
CPP	14.379	1	14.379	4.413	.042*	.093
Degree of Embodiment × CPP	.273	1	.273	.084	.773	.002
Error	140.112	43	3.258			
Total	910.00	47				

$R^2 = .261$, adj. $R^2 = .210$

Design: Intercept + Degree of embodiment + CPP + Degree of embodiment × CPP

* $p < .05$, ** $p < .01$

6.3 Programming Accuracy and Efficiency

When investigating accuracy on the programming tasks, the significance of CPP was observed $F(1,62) = 18.438$, $p = .000$, partial $\eta^2 = .23$. The post hoc test indicated that the full-embody with CPP group's programming accuracy ($n = 20$, $M = 3.00$, $SD = 1.56$) was significantly higher than that of the full-embody without CPP group ($n = 21$, $M = 1.33$, $SD = 1.11$) with a mean difference of 1.66, $p = .006$. Within the low-embody groups, the group with CPP ($n = 11$, $M = 3.36$, $SD = 1.96$) produced more accurate programming than the low-embody group without CPP ($n = 14$, $M = 1.64$, $SD = 1.73$) with a mean difference of 1.72, $p = .037$, with $F(3,62) = 6.641$, $p = .001$. Finally, the programming accuracy score for full-embody with CPP and low-embody with CPP was significantly higher than for full-embody without CPP and low-embody without CPP (Table 4).

With regard to programming efficiency, a nonparametric Mann–Whitney test was used because Levene's test failed to support the homogeneity of variance of the frequency data. The results indicated that the presence of CPP was significant whereas the impact of the degree of embodiment was insignificant. Specifically, the with CPP group ($M = 0.612$, $Mdn = 0.00$) showed significantly greater programming efficiency skills, $U = 354.00$, $p = .001$, $r = 0.40$. A Kruskal–Wallis test was conducted to evaluate differences among the four conditions and a significant result was observed χ^2 (3, $n = 66$) = 11.091, $p = .011$. The full-embody with CPP group and the low-embody with CPP group showed a higher mean rank than the groups without CPP.

Table 4 Two-way analysis of variance on programming task scores by embodiment and CPP factor

Source	SS	Df	MS	F	Sig.	Partial η^2
Degree of embodiment	1.743	1	1.743	.728	.397	.012
CPP	44.141	1	44.141	18.438	.000**	.229
Degree of embodiment × CPP	.011	1	.011	.005	.946	.000
Error	148.426	62	2.394			
Total	528.000	66				

$R^2 = .243$, adj. $R^2 = .207$

Design: Intercept + Degree of embodiment + CPP + Degree of embodiment × CPP

* $p < .05$, ** $p < .01$

Overall, the presence of CPP was significant for programming accuracy and efficiency, regardless of the degree of embodiment. A more accurate and efficient programming outcome was found in the full-embody with CPP and low-embody with CPP groups, confirming the transfer of programming concepts learning through intervention activities.

7 Discussion

This experiment produced interesting findings and implications for further research. Specifically, there were significant main effects for a greater degree of embodiment on post-test numeracy, magnitude comparisons, applied number line estimation, and delayed test items. Conversely, the presence of CPP was significant on the delayed test and programming performance.

A significant impact of the embodied activities was found in basic numeracy abilities on the post-test (i.e., addition, subtraction, number order, and magnitude comparisons), other arithmetic problems, and number line estimation in the applied item, indicating that students' application of learned knowledge differed by group. When given the novel number line in the applied and delayed test, students in the full-embody group demonstrated improved arithmetic ability and more flexible understanding of the number line. These results strengthen the belief that learners' participation through active interaction within the physical world promotes concrete sensorimotor experiences, which, in turn, support the learning of abstract concepts (Abrahamson and Howison 2010; Abrahamson and Lindgren 2014; Bamberger and DiSessa 2003; Lakoff and Johnson 1999).

With regard to retention and transfer, results of the delayed test suggest differences in number line linearity depending on the presence of CPP (Johnson-Glenberg et al. 2014). Children who were asked to provide commands to a surrogate by decomposing steps to solutions developed robust learning and improved their number line estimation skills. Ericsson and Simon (1980) discussed the processes underlying verbalization, arguing that it is a direct articulation or explication of information processing. The CPP requiring students to produce concurrent verbalization to manipulate a surrogate forced them to form an internal representation of the moves, distances, and number sense. Moreover, the full-embody activity combined with CPP promoted transfer to novel number line, arithmetic, and PN tasks whereas the low-embodied activity without CPP failed to do so.

In addition to the development of mathematics understanding, the main goal of this study was to promote computational thinking, a fundamental cognitive skill for programming learning that permeates the STEM world. While students in all groups received minimum instruction about Scratch Jr. functions (e.g., coding block drag and drop, how to navigate, and how to simulate), the impact of CPP on students' programming accuracy and proficiency was significant. The second programming task developed to measure programming accuracy and proficiency skills was challenging for novice learners since diverse solutions may be designed based on programming fluency and utilization of conditional blocks. Programming accuracy and programming efficiency skills were significantly affected by the presence of CPP. Regardless of the degree of embodiment, groups practicing computational perspectives demonstrated abstraction, sequential thinking, and pattern recognition skills. Considering the students' minimum prior knowledge about programming and the limited guidance provided, their problem-solving skills with programming concepts reflect the effectiveness of the learning activities. These results indicate that taking the perspective that is unique to a computer programmer or a scientist

(Berland and Wilensky 2015) exposed students to computational thinking such as task decomposition, sequential thinking, procedural thinking, and commanding skills to operate a surrogate. The explicit nature of CPP maximizes students' understanding of number line, numeracy, number sense, and programming concepts, thus leading to a robust understanding of mathematics and programming concepts, as discussed in Papert (1980) and Pea and Kurland (1984).

In short, CPP-engaged embodied activities delivered before the introduction of the first programming task promoted computational thinking that benefited mathematics as well as programming learning. The authors find it possible to design coherent classroom activities that effectively embed computational perspectives into the existing mathematics and programming curricula in an interdisciplinary manner to improve students', particularly underrepresented groups, learning within both mathematics and programming.

8 Conclusion

This study produced important findings and implications for further research in the search for an effective K-12 curriculum to foster computational thinking and programming skills targeting underrepresented groups. If programming concept is combined with other domains with computational perspectives, it can provide an effective setting for guiding students' cognitive processing to focus on how to think rather than on what to think, which is fundamental to computational thinking (Clements and Gullo 1984).

Considering the characteristic of target learner group, the goal was not only to focus on effective learning activities, but also to put more emphasis on how to support underrepresented groups or under-resourced schools. Thus, the study highlighted the importance of activities students performed rather than relying solely on the use of programming software.

When developing the interventions, the study aimed to lower dependence on technological tools and rely more on activities that can be easily plugged into real-world contexts. In the present study, the programming activity was not part of the intervention, but served as a measure of programming accuracy and proficiency. All participants received equal opportunity and time to play on an iPad, meaning that the presence or quality of the tool was not a distinguishing factor. Therefore, the significant differences found across groups were due to differences in learning activities, highlighting the importance of appropriate learning activities that are not vulnerable to the quality or quantity of technological tools.

Designing a learning environment that promotes explicit thinking processes is the key in deciding how to introduce programming in early education and assist students in learning to program. A learning environment where students engage full-body movements and CPP leads to greater engagement and improved learning outcomes. Compared to introducing digital tools without providing opportunities to develop learners' own understanding of programming concepts, it is more powerful to prepare students in appropriate learning activities that support computational perspectives and practices and then expose them to digital tools.

The proposed activities exhibit the potential that computational practices and perspectives can be developed through self-discovery. We propose that future research focus on supporting students' problem-solving skills in designing self-exploratory instructional activities rather than teaching technical rules and skills. Moreover, it is necessary to extend

the current pedagogical design and results from an after-school program to formal educational settings to inform computational education on a greater scale.

References

Abrahamson, D., & Howison, M. (2010). *Embodied artifacts: coordinated action as an object-to-think with.* Denver, CO: In annual meeting of the American Educational Research Association.

Abrahamson, D., & Lindgren, R. (2014). Embodiment and embodied design. In R. K. Sawyer (Ed.), *The Cambridge handbook of the learning sciences* (2nd ed., pp. 358–376). Cambridge: Cambridge University Press. doi:10.1017/CBO9781139519526.022.

Abrahamson, D., & Trninic, D. (2015). Bringing forth mathematical concepts: Signifying sensorimotor enactment in fields of promoted action. *ZDM Mathematics Education, 47*(2), 295–306.

Alimisis, D. (2013). Educational robotics: Open questions and new challenges. *Themes in Science and Technology Education, 6*(1), 63–71.

Bamberger, J., & DiSessa, A. (2003). Music as embodied mathematics: A study of a mutually informing affinity. *International Journal of Computers for Mathematical Learning, 8*(8), 123–160. doi:10.1023/B:IJCO.0000003872.84260.96.

Barsalou, L. W. (2008). Grounded cognition. *Annual Review of Psychology, 59*(1), 617–645. doi:10.1146/annurev.psych.59.103006.093639.

Barsalou, L. W., Niedenthal, P. M., Barbey, A. K., & Ruppert, J. A. (2003). Social embodiment. *Psychology of Learning and Motivation, 43,* 43–92.

Berland, M., & Wilensky, U. (2015). Comparing virtual and physical robotics environments for supporting complex systems and computational thinking. *Journal of Science Education and Technology, 24*(5), 628–647. doi:10.1007/s10956-015-9552-x.

Bers, M. U. (2008). Using robotic manipulatives to develop technological fluency in early childhood. In O. N. Saracho & B. Spodek (Eds.), *Contemporary perspectives on science and technology in early childhood education, LAP 105–225*. Greenwich, CT: Information Age Publishing Inc.

Bers, M. U. (2010). The TangibleK robotics program: Applied computational thinking for young children. *Early Childgood Research & Practice, 12*(2), 1–20. Retrieved from http://ecrp.uiuc.edu/v12n2/bers.html.

Black, J. B., Segal, A., Vitale, J., & Fadjo, C. (2012). Embodied cognition and learning environment design. *Theoretical Foundations of Learning Environments, 2,* 198–223.

Booth, J. L., & Siegler, R. S. (2008). Numerical magnitude representations influence arithmetic learning. *Child Development, 79*(4), 1016–1031.

Brophy, S., Klein, S., Portsmore, M., & Rogers, C. (2008). Advancing engineering education in P-12 classrooms. *Journal of Engineering Education, 97*(3), 369–387.

Burke, Q. (2012). The markings of a new pencil: Introducing programming-as-writing in the middle school classroom. *Journal of Media Literacy Education, 4*(2), 121–135.

Chan, M. S., & Black, J. B. (2006). Direct-manipulation animation: Incorporating the haptic channel in the learning process to support middle school students in science learning and mental model acquisition. In *Proceedings of the 7th International Conference on Learning Sciences* (pp. 64–70). Bloomington, IN.

Clements, D. H., & Battista, M. T. (1989). Learning of geometric concepts in a Logo environment. *Journal for Research in Mathematics Education, 20*(5), 450–467.

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

Clements, D. H., & Sarama, J. (2002). The role of technology in early childhood learning. *Teaching Children Mathematics, 8*(6), 340–343.

Einhorn, S. (2011). *Micro-worlds, computational thinking, and 21st century learning.*[White paper] Retrieved from http://el.media.mit.edu/logofoundation/resources/papers/pdf/computational_thinking.pdf.

Ericsson, K. A., & Simon, H. A. (1980). Verbal reports as data. *Psychological Review, 87*(3), 215–251.

Fadjo, C. L. (2012). *Developing Computational Thinking through Grounded Embodied Cognition* (Unpublished doctoral dissertation). Columbia University, NY.

Fadjo, C. L., Hallman Jr., G., Harris, R., & Black, J. B. (2009). *Surrogate embodiment, mathematics instruction and video game programming*. Paper presented at the World Conference on Educational Media and Technology, Honolulu, HI. http://www.editlib.org/p/31876.

Fadjo, C., Lu, M., & Black, J. B. (2009). *Instructional embodiment and video game programming in an after school program*. Paper presented at the World Conference on Educational Media and Technology, Honolulu, HI. http://www.editlib.org/p/32064.

Feurzeig, W., Papert, S., & Lawler, B. (2011). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments, 19*(5), 487–501. doi:10.1080/10494820903520040.

Fryer, W. A. (2014). *Hopscotch challenges: Learn to code on an iPad!*. Retrieved from http://publications.wesfryer.com/index.php/archive/article/view/53.

Glenberg, A. M. (2008). Toward the integration of bodily states, language, and action. In G. R. Semin & E. R. Smith (Eds.), *Embodied grounding: Social, cognitive, affective, and neuroscientific approaches* (pp. 43–70). New York: Cambridge University Press.

Glenberg, A. M. (2010). Embodiment as a unifying perspective for psychology. *Wiley Interdisciplinary Reviews: Cognitive Science, 1*(4), 586–596.

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.

Hallman, G., Paley, I., Han, I., & Black, J. (2009). Possibilities of haptic feedback simulation for physics learning. In *Proceedings of world conference on educational multimedia, hypermedia and telecommunications* (pp. 3597–3602). Honolulu, HI.

Huang, S. C., Vea, T., & Black, J. (2011). Learning classic mechanics with embodied cognition. In *Proceedings of world conference on e-learning in corporate, government, healthcare, and higher education* (pp. 209–215). Chesapeake, VA.

Hughes, M., & Macleod, H. (1986). Using logo with very young children. In R. Lawler, B. du Boulay, M. Hughes, & H. Macleod (Eds.), *Cognition and computers: Studies in learning* (pp. 179–219). Chichester: Ellis Horwood.

International Society for Technology in Education. (2011). *Operational definition of computational thinking for K–12 education*. Available at http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf?sfvrsn=2.

Johnson-Glenberg, M. C., Birchfield, D. A., Tolentino, L., & Koziupa, T. (2014). Collaborative embodied learning in mixed reality motion-capture environments: Two science studies. *Journal of Educational Psychology, 106*(1), 86–104.

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.

Kurland, D. M., & Pea, R. D. (1985). Children's mental models of recursive logo programs. *Journal of Educational Computing Research, 1*(2), 235–243.

Lakoff, G., & Johnson, M. (1999). *Philosophy in the flesh: The embodied mind and its challenge to western thought*. New York: Basic Books.

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

Lindgren, R. (2014). Getting into the cue: Embracing technology-facilitated body movements as a starting point for learning. In V. R. Lee (Ed.), *Learning technologies and the body: Integration and implementation in formal and informal environment* (pp. 39–54). New York, NY: Routledge.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41,* 51–61. doi:10.1016/j.chb.2014.09.012.

Matarić, M. J., Koenig, N., & Feil-Seifer, D. (2007). Materials for enabling hands-on robotics and STEM education. In *AAAI spring symposium on robots and robot venues: Resources for AI education* (pp. 99–102). Stanford, CA.

National Research Council. (2012). *A framework for K-12 science education: Practices, crosscutting concepts, and core ideas*. Washington, DC: The National Academies Press.

Papert, S. (1972). Teaching children thinking. *Programmed Learning and Educational Technology, 9*(5), 245–255.

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

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology, 2*(2), 137–168.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., et al. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60–67.

Robinson, M. A., & Uhlig, G. E. (1988). The effects of guided discovery Logo instruction on mathematical readiness and visual motor development in first grade students. *Journal of Human Behavior and Learning, 5,* 1–13.

Schwartz, D. L., & Black, J. B. (1996). Shuttling between depictive models and abstract rules: Induction and fallback. *Cognitive Science, 20*(4), 457–497.

Siegler, R. S., & Opfer, J. E. (2003). The development of numerical estimation evidence for multiple representations of numerical quantity. *Psychological Science, 14*(3), 237–250.

Wilson, M. (2002). Six views of embodied cognition. *Psychonomic Bulletin & Review, 9*(4), 625–636. doi:10.3758/BF03196322.

Wilson, A. D., & Golonka, S. (2013). Embodied cognition is not what you think it is. *Frontiers in Psychology, 4,* 1–13. doi:10.3389/fpsyg.2013.00058.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. doi:10.1145/1118178.1118215.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717–3725.