Computers & Education 72 (2014) 145–157

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

# Computational thinking and tinkering: Exploration of an early childhood robotics curriculum

Marina Umaschi Bers, Louise Flannery, Elizabeth R. Kazakoff*, Amanda Sullivan

*Tufts University, Medford, MA, USA*

ARTICLE INFO

*Article history:*
Received 30 January 2013
Received in revised form
22 October 2013
Accepted 29 October 2013

*Keywords:*
Elementary education
Interactive learning environments
Pedagogical issues
Teaching/learning strategies
robotics
Programming
Early childhood

ABSTRACT

By engaging in construction-based robotics activities, children as young as four can play to learn a range of concepts. The TangibleK Robotics Program paired developmentally appropriate computer programming and robotics tools with a constructionist curriculum designed to engage kindergarten children in learning computational thinking, robotics, programming, and problem-solving. This paper documents three kindergarten classrooms' exposure to computer programming concepts and explores learning outcomes. Results point to strengths of the curriculum and areas where further redesign of the curriculum and technologies would be appropriate. Overall, the study demonstrates that kindergartners were both interested in and able to learn many aspects of robotics, programming, and computational thinking with the TangibleK curriculum design.



## 1. Introduction

For decades, early childhood (preschool to grade two) curricula have focused primarily on literacy and math, especially with the educational reforms of No Child Left Behind (Zigler & Bishop-Josef, 2006). However, there has been some recent attention to science, technology, engineering, and math (STEM) learning for young children (Gelman & Brenneman, 2004; Sesame Workshop, 2009; White House, 2011). Furthermore, new technology learning standards and best practices for integrating technology into early childhood education have been developed (Barron et al., 2011; International Society for Technology in Education (ISTE), 2007; NAEYC & Fred Rogers Center for Early Learning and Children's Media, 2012; U.S. Department of Education, 2010). Of note, the technology policy statement from NAEYC & Fred Rogers Center for Early Learning and Children's Media (2012) provides a guide for early childhood education professionals in using interactive digital technologies in balanced and developmentally appropriate ways. It addresses important issues related to using digital technology with children ages three–eight years, including the needs for technology use to serve the needs of the children, and for educators to be able to understand, evaluate, and integrate developmentally appropriate technologies in their classrooms. However, there is little research on computer programming specifically for early childhood, the subject this paper explores.

As new devices, from smartphones and tablet computers to electronic learning toys, find new audiences with increasingly young children, challenging question arise about how to define developmentally appropriate activities and content for children of different ages. While the majority of research on robotics and programming in education focuses on later schooling, teaching these subjects during foundational early childhood years can be an engaging and rewarding experience for young learners (Bers, 2008). Previous research has shown that children as young as four–six years old can build and program simple robotics projects (Bers, Ponte, Juelich, Viera, & Schenker, 2002, pp. 123–145; Cejka, Rogers, & Portsmore, 2006; Kazakoff, Sullivan, & Bers, 2012; Perlman, 1976, p. 260; Wyeth, 2008) as well as learn powerful ideas from engineering, technology, and computer programming while also building their computational thinking skills (Bers, 2008). Robotic manipulatives allow children to develop fine-motor skills and hand–eye coordination while also engaging in collaboration and teamwork. Additionally, robotics can provide a fun and playful way for teachers to integrate academic content with the creation of

* Corresponding author. DevTech Research Group, Eliot Pearson Department of Child Development, 105 College Ave., Medford, MA 02155, USA. Tel.: +1 617 347 5746.
*E-mail address:* Elizabeth.Kazakoff@Tufts.edu (E.R. Kazakoff).

0360-1315/$ – see front matter 
http://dx.doi.org/10.1016/j.compedu.2013.10.020

meaningful projects. Through robotics, young children can experiment with concepts of engineering as well as storytelling by creating narrative contexts for their projects (Bers, 2008). By engaging in these types of robotics projects, young children play to learn while learning to play in a creative context (Resnick, 2003).

Computers offer new ways of representing and interacting with information and an entirely new category of "objects to think with" (Papert, 1980). In the form of programmable and interactive robots, computers can become powerful learning tools. Robotics offers children the opportunity to engage with content from the domain of computer science, practice problem-solving skills, and work on fine-motor skills and eye–hand coordination. The TangibleK Robotics Program, a design-based research initiative now in its fifth year, has paired developmentally appropriate programming and robotics tools with a curriculum to engage kindergartners in learning computational thinking, robotics and programming concepts, as well as problem-solving and reasoning. The goal of this paper is to present young children's learning outcomes on computer programming concepts as taught through the TangibleK curriculum in order to highlight the potential for learning of integrating computer programming and robotics into the early childhood classroom.

### 1.1. Theoretical framework: constructionism and positive technological development

The theoretical approach used for designing the educational intervention and curriculum and for integrating the TangibleK Robotics Program into early childhood classrooms incorporates elements from Papert's (1980) constructionist framework, which states that children can learn deeply when they build their own meaningful projects in a community of learners and reflect carefully on the process.

Papert's (1980) *constructionism* is rooted in Piaget's (1954) *constructivism* – which conveys the idea that the child actively builds knowledge through experience – and the related "learn-by-doing" approach to education. While Piaget's (1954) theory was developed to explain how knowledge is constructed in an individual's mind, Papert (1980) expands on it to focus on the ways that internal constructions are supported by constructions in the world, including through the use of computers and robotics. A constructionist teaching approach provides children the freedom to explore their own interests through technologies (Bers, 2008) while investigating domain-specific content learning and also exercising meta-cognitive, problem-solving, and reasoning skills (e.g., Clements & Gullo, 1984; Clements & Meredith, 1992). Papert (1980) discussed that well-designed constructionist activities have embedded in them 'powerful ideas', central concepts within a domain that are both epistemological and personally useful, interconnected with other disciplines, and have roots in intuitive knowledge that a child has internalized over a long period of time (Bers et al., 2002; Papert, 1980). An idea may be considered powerful to the degree that it is useful in building and extending further knowledge (Papert, 2000). The robotics curriculum described in this paper is composed of powerful ideas from the domains of computer science and engineering (e.g., the engineering design process, debugging, robotic motion and sensing, using programming instructions, control flow by sequence, control flow by specific instructions).

Classroom activities designed to impact learning outcomes and cognitive growth, also have an impact on (and are influenced by) children's social, emotional, and moral development. As a framework to guide the design and implementation of a robotics curriculum that also focuses on these dimensions of the child, Bers' (2010, 2012) Positive Technological Development (PTD) was utilized. PTD takes into consideration the learning environment and pedagogical practices, as well as cultural values and rituals, which mediate teaching and learning (Bers, 2008; Rogoff, Goodman Turkanis, & Bartlett, 2001). The educational experience proposed by the presented robotics curriculum was structured using the PTD framework to encourage six behaviors, which in turn foster the development of beneficial core cognitive and social traits. Specifically, engaging in content generation, creative design and problem-solving, collaboration, communication, choices of conduct, and community-building may lead to a sense of competence and confidence, the ability to connect with and care about others, contribution to entities outside the self, and moral character (Bers, 2010, 2012). For instance, by iteratively planning and revising a robotics project in a supportive environment, children may gain confidence in their abilities to learn and solve problems. Alternatively, discussions of how to share limited resources fairly amongst the class are opportunities for positive moral development.

### 1.2. Learning through computer programming

Embedded in the exploration of computer programming and robotics, the TangibleK curriculum also fosters *computational thinking*. This term has been defined in many ways and encompasses a broad and somewhat debated range of analytic and problem-solving skills, dispositions, habits, and approaches used in computer science (Barr & Stephenson, 2011; International Society for Technology Education and The Computer Science Teachers Association, 2011; Lee et al., 2011). The TangibleK curriculum specifically fosters computational thinking skills such as: problem representation; systematicity in generating and implementing solutions; exploring multiple possible solutions; problem-solving on multiple levels – from approaching the overall challenge to "debugging" or trouble-shooting specific difficulties with a given solution's implementation; productive attitudes toward "failure" and misconceptions uncovered along the route to a successful project; and strategies for approaching open-ended and often difficult problems. Such skills are of general applicability beyond robotics and computational thinking.

### 1.3. The TangibleK Robotics Program

The TangibleK Robotics Program, whose design is informed by the theoretical frameworks of constructionism and PTD, has iteratively implemented and assessed a set of programming and robotics tools, curricula, and pedagogical approaches in close collaboration with hundreds of children and dozens of teachers over the course of five years. The research goals of the TangibleK Robotics Program are to:

1) Provide an evidence-based description of young children's learning trajectories in computational thinking and capacity to understanding computer programming and robotics concepts when given developmentally appropriate materials,
2) Develop and test an early childhood curriculum to teach developmentally appropriate concepts from computer programming and robotics to children in kindergarten through second grade,
3) Investigate the design features of the programming interface and the mediating role interface design plays in learning to program.

This paper addresses the first of these goals, to describe young children's learning trajectories in computational thinking and capacity to understand computer programming and robotics concepts. This understanding will allow further revision to the TangibleK curriculum.

The TangibleK Robotics Project makes use of commercially available robotics construction kits and the CHERP (Creative Hybrid Environment for Robotics Programming) language to give behaviors to the robotic constructions (Bers, 2008; Bers & Horn, 2010; Horn et al., 2011; Kazakoff & Bers, 2012; Kazakoff, Sullivan, & Bers, 2012). CHERP is a hybrid tangible and graphical computer language designed to provide young children with an engaging introduction to computer programming in a developmentally appropriate way. The software allows children to create programs to control their robots from tangible wooden blocks and/or graphical, on-screen icons. The design of CHERP avoids the technical and syntax-related challenges of text-based programming languages. Furthermore, the hybrid interface allows children to choose the interface that best suits their changing preferences as physical abilities, perceived social appeal, and the level of challenge of the activity at hand evolve (Horn et al., 2011), because both tangible and graphical interfaces can represent the same concepts.

The TangibleK curriculum introduces increasingly complex powerful ideas from computer science in a robotics context in a structured, developmentally appropriate way. The powerful ideas from computer science addressed in this curriculum include: the engineering design process and debugging (trouble-shooting), robotic motion and sensing, and three aspects of programming: choosing the correct programming instructions, controlling the flow of actions by sequencing the action instructions accordingly, and controlling the flow of actions by using special control flow instructions. Section 2.3 contains more detailed definitions of each powerful idea. In addition to the concrete robotics and programming concepts and skills introduced in each activity, skills such as observation, reflection, and decomposition of complex processes are interwoven throughout the curriculum.

The curriculum, which takes approximately 20 h of classroom work, includes six structured 60- to 90-min activities and a culminating interdisciplinary project. All the activities focus on building and programming a robotic vehicle to accomplish a particular goal. Each lesson addresses one or more powerful idea(s) within the context of a narrative theme. The six lesson activities and their embedded content are as follows:

- Lesson 1: The Engineering Design Process

Children build sturdy, non-robotic vehicles to transport toy people on a floor map. Children apply the stages of the Engineering Design Process to plan, test, and improve their vehicles.

- Lesson 2: Robotics

Children share and learn ideas about what robots are and are not. They explore robotic parts by designing and building their own robots. They learn to appropriately connect robotic parts (e.g., snap-together wires and motors) to make a robot that moves.

- Lesson 3: Choosing and Sequencing Programming Instructions

In this activity, children program their robots to dance the "Hokey-Pokey" by choosing relevant instructions and putting them in the correct order or sequence.

- Lesson 4: Looping Programs (Control Flow Instructions 1)

Children use "repeat" instructions to program their robots to move forward forever. Next, they program their robot to move forward only a particular number of times to reach a fixed location.

- Lesson 5: Sensors

Children use light sensors to program their robots to turn its light on when it is dark out and vice versa. They draw comparisons between robotic sensors and the five human senses.

- Lesson 6: Branching Programs (Control Flow Instructions 2)

Children are introduced to a pair of *conditional* control flow instructions, "If" and "If Not", which are also used with a sensor to make programs that incorporate environmental conditions into the robot's behavior.

In addition to the structured activities described above, the TangibleK curriculum includes songs, games, and free-play with the robotics and programming materials in order to foster a playful learning environment for children. For example, in Lesson 3, children sing and dance the "Robot Hokey-Pokey" and play Simon Says with the CHERP programming commands to recall and apply the programming instructions. Throughout the 20 h curriculum, children have ample opportunity to freely build and design with the robotics materials and to create their own CHERP programs, beyond those that are set forth in each of the structured lessons.

After completing the six lessons described above, each classroom embarks on a culminating, interdisciplinary project, which invite children to apply the now familiar powerful ideas to a particular theme or context. The teacher decides on a theme drawn from other subjects studied during the year, and each child chooses a challenge within this theme. Past classrooms have selected topics such as animal behaviors, vehicles that help the community, or "Who Am I?" Children created projects representing snakes that slither, recycling trucks that collect refuse, and sewing needles that travel back and forth through fabric, among many others. The projects allow children to demonstrate the powerful ideas they learned over the six activities as well as to apply them and continue learning about them in a new context.

Having introduced an overview of the TangibleK Robotics Program, including its technological, curricular, and theoretical components, we now present a study of three kindergarten classrooms in which the TangibleK Robotics Program was implemented. The following sections report the distribution of achievement scores children attained on selected computer programming concepts and skills tied to the

powerful ideas listed above. Achievement scores form the basis on which to discuss the curriculum structure and content and consider the implications for understanding children's early learning trajectories of computational concepts and for further adaptation of the curriculum.

## 2. Study design

Within the design-based research tradition of iterative testing, analysis, and refinement of an intervention, (see, e.g., Cobb, Confrey, diSessa, Lehrer, & Schauble, 2003), the TangibleK Robotics Program has spent five years exploring what children are capable of learning and accomplishing in the domains of robotics and programming. The study described in this paper examines how successfully children learned the core concepts (powerful ideas) of robotics and programming in the TangibleK curriculum. The study took place during the fourth year of the overall project, following piloting and refinement of the software and curriculum in a range of settings, from classrooms and after-school/summer programs to the research lab. The extensive testing, exploration, and refinement of the preceding study iterations also laid a foundation for understanding how young children learn and think about core concepts of programming and robotics. For instance, several of the curricular activities were simplified to enable better focus on the target concepts; movement games and songs were added to the curriculum to engage children in multiple modes of understanding concepts and to provide reinforcement for basic knowledge. In addition, some of the programming icons were revised to use more familiar imagery for children.

### 2.1. Participants

Each of the three teachers involved in this study volunteered to participate following email notification of the opportunity to principals of a limited number of schools in the Greater Boston area. All children in each classroom participated in the curriculum, but each family had the option to allow or decline data collection. According to school community needs, consent materials were available in English, Portuguese, and Spanish.

Children in the study attended one of three Greater Boston area kindergarten classrooms, two of which were at a public urban school, and one at a private suburban school. From a total of 63 children enrolled in the three classes during the study, 53 are included in data analysis. Children were included in data analysis unless they missed more than one activity or if data was not collectible for more than one activity. Attrition was due to typical classroom absences as well as the difficulty of collecting data with limited researchers in a bustling classroom environment.

Classroom 1, a kindergarten in an independent, K-8, religious-based, private school in a suburb of Boston, MA, had 23 children, 18 of whom are included in data analysis. The student population at this school was 97% White, 1% as Asian, 1% as Black, 1% Hispanic (http://nces.ed.gov/globallocator/). Of the children in the kindergarten class, 50% were male and 50% were female. They ranged from ages 4.9 to 6.2 years at the start of the study, with a median age of 5.6 years. The only kindergarten classroom at this school, this class was taught by a male teacher with seven years of teaching experience, who, on a scale from 1 (none) to 5 (expert), rated his computer experience as 5, programming experience as 3, and robotics experience as 1.

Classrooms 2 and 3 were located at the same urban K-8 school (NCLB Level 3), located just outside of Boston, MA. The makeup of this school during the 2010–2011 school year was 38.9% White, 36.3% Hispanic, 16.2% African American, 7.0% Asian American, and 1.7% multi-race. The school was comprised of 41.1% English-Language Learners and 64.4% of students were classified as low income (Massachusetts Department of Education, 2006).

A female teacher with six years of teaching experience taught Classroom 2. She rated her computer experience as a 4, robotics experience as a 2, and programming experience as a 2. This classroom had 19 children enrolled, 17 of whom are included in the data analysis. Of those 17 children, 59% were male and 41% were female. At the start of the curriculum, the children in this classroom ranged in age from 4.9 to 6.2 years old, with a median age of 5.6 years.

A female teacher with 15 years of teaching experience taught Classroom 3. She also rated her computer experience as a 4, robotics experience as a 2, and programming experience as a 2. The data analysis includes 18 of 21 children enrolled in this classroom. Of the 18 participants, 44% were female and 56% were male. The children's ages at the start of the curriculum in Classroom 3 ranged from 5.6 to 6.5 years, with a median age of 6.0 years old.

The overall age range for the 53 children included in data analysis was 4.9–6.5 years, and their average age at the start of the curriculum was 5.7 years old. Over the three classrooms as a whole, 45% of the children were female and 55% male. The participants in this study are thought to be generally representative of the general kindergartners population, as the sample includes both public and private school students, both male and female teachers, a fairly even proportion of male and female students, and, as described above, a diverse range of ethnic and socioeconomic backgrounds, particularly at to the participating public school.

### 2.2. Procedure

Each classroom's head teacher and all research assistants (nearly 20 research collaborators in total) received training to prepare them for teaching or assisting the robotics curriculum and participating in the research and data collection. The high number of assistants was needed for two reasons. First, a low student-to-adult ratio in each lesson ensured adequate observation and documentation of students' work. Secondly, most research assistants had limited availability across the full set of study sessions. Therefore, attention was given to all collaborators' to ensure they received careful and detailed training. The 3-h introductory training covered technical, curricular, and pedagogical aspects of the program including how to use the CHERP programming language and LEGO® robotics kits as well as activity content and training on the structure and the teaching approach framed by the PTD model presented earlier. The training also included explanation and examples of how to score children's work in each activity according to a scale of understanding levels, described below.

The teachers then implemented the TangibleK curriculum in their own classrooms with technical support from trained research assistants. Two teachers used the curriculum with the whole class working together. The third teacher worked with half of the class at a time, finishing the entire curriculum with one group before starting it with the other. Each curricular activity took one to two 60–90 min session(s). The teacher introduced key concepts and the day's activity in a whole-group setting along with a short song or game to reinforce the

concepts. As mentioned earlier, in Lesson 3, each class sang and danced the "Hokey-Pokey" before programming their robots to do this dance. Additionally, the game "Simon Says" was often used in Lessons 3–6 to reinforce the CHERP programming instructions and their corresponding robotic actions. After the whole-group activities, children built and/or programmed their own robotic vehicles. The children worked independently on their projects but sat in groups of four and received support as needed from the research assistant or classroom teacher at their group while also interacting with their peers. With the variety of coders evaluating children's work, we systematically accounted for potential intercoder differences by varying which adult worked with which children during each lesson. Each session's work ended with a group discussion for children to share progress, questions, and successful strategies, and for the teacher to help reinforce the core robotics and programming concepts and the engineering design process.

To assess learning outcomes after each activity, research assistants evaluated the robot and/or program made by each child. They assessed the child's level of understanding of selected core concepts as seen by successful application of the concepts in the robot or program. If needed, they also talked with children to gain more information about their work and understandings. By examining, for instance, the child's program for correct selection and sequencing of action instructions or proper use of the "repeat" instruction, research assistants scored each child's achievement of the core goals of the lesson using the following 6-point Likert scale designed to document the thoroughness of the child's understanding and application of activity-specific concepts and skills as well as their use of general problem-solving skills. A score of 4 or higher was defined as the target level of achievement.

5 Complete achievement of the goal, task, or understanding
4 Mostly complete achievement of the goal, task, or understanding;
3 Partially complete achievement of the goal, task, or understanding;
2 Very incomplete achievement of the goal, task, or understanding;
1 Did not complete the goal, task, or understanding;
0 Did not attempt/Other.

In each lesson, children were scored on multiple concepts using this Likert scale. For example, in Lesson 3, children programmed their robots to dance the "Hokey-Pokey" by 1) choosing the correct instructions (a skill referred to here as correspondence) and 2) putting the instructions in the correct order (sequencing). The concepts of sequencing and correspondence are described in more detail in Sections 2.3.2 and 2.3.3. As an illustration of the general scale, children received one point on the correspondence scale for each programming instruction that correctly matched a line of the song. Below are examples of children's programs that were scored at each level of correspondence in Lesson 3:

5 Begin, Forward, Backward, Forward, Shake, Spin, End (all correct);
4 Begin, Forward, Forward, Forward, Shake, Spin, End (second Forward should be Backward);
3 Begin, Forward, Backward, Shake, End (missing Forward and Spin);
2 Begin, Shake, Spin, End (missing Forward, Backward, Forward)
1 Begin, Shake, End (missing Forward, Backward, Forward, and Spin).
0 Despite assistance and prompting, the child did not attempt Hokey-Pokey task.

These same programs also received a 0–5 score for sequencing in Lesson 3.

### 2.3. Variables examined

To examine children's growing computational thinking ability throughout implementation of the TangibleK curriculum, four key variables were observed and assessed: debugging, correspondence, sequencing, and control flow.

#### 2.3.1. Debugging

When faced with a difficult problem or task, children (and adults) are often unable to determine a suitable solution on the first attempt. In these situations, "debugging" skills can be helpful. Debugging, or trouble-shooting, is a form of problem-solving used in the fields of engineering and computer science. It encompasses four steps: 1) To debug a problem, the child must first recognize that something is not working or not meeting the stated goal. For example, a child programming her robot to dance the Hokey-Pokey in Lesson 3 watches her program running and realizes that the robot does not "shake it all about". 2) In step 2 of the debugging process, children either decide to keep their original goal or switch to an appropriate alternative. This child might continue to pursue the original plan of making the robot dance all the parts of the Hokey-Pokey, or, as is common at this age, she might come up with an alternative, such as having their robot do a different dance. 3) The third stage of debugging is generating a hypothesis as to the cause of the problem. The child in our example may hypothesize that the program is missing an instruction that would make their robot shake. 4) Finally, the last aspect of debugging is attempting to solve the problem. The child might put a "Shake" block in different positions in the program until the program fully matches the song. Debugging skills are not limited to the arena of engineering and computer science. Previous research has found that children can acquire and transfer debugging skills to activities outside of the programming context with appropriate support, including explicit instruction (Klahr & Carver, 1988; Salomon & Perkins, 1987).

The steps of the debugging process are a critical component of the Engineering Design Process, which refers to the cyclical or iterative process engineers use to design an artifact in order to meet a need (Massachusetts Department of Education, 2006). As defined by the MA curriculum frameworks, its steps include: identifying a problem, looking for ideas for solutions and choosing one, developing a prototype, testing, improving, and sharing solutions with others (see Fig. 4). The steps of testing and improving, which require debugging, are particularly important in establishing a learning environment where failure – rather than immediate success – is expected and seen as necessary for learning. With the Engineering Design Process, children are not expected to "get it right" the first time.

In the TangibleK curriculum, debugging and the Engineering Design process were first introduced in Lesson 1, and the concepts and skills were applied throughout the rest of the curriculum. Children were assessed on their ability to apply the four core aspects of debugging (described above) in each lesson and final project.

#### 2.3.2. Correspondences between actions and instructions

A program is a sequence of instructions that a computer (in this case, a robot) acts out in an order specified by the programmer (Stair & Reynolds, 2003). Each instruction has a specific meaning, and the order of the instructions leads to the robot's overall actions. Making correspondences between actions and instructions encompasses the understanding that each programming instruction represents a specific action carried out by the robot.

Another way to understand the process of correspondence is to frame it with the notion of symbols, a core concept that children are learning in kindergarten in both math and literacy. Each programming instruction is a *symbol* for the action the robot will carry out. In order to program a robot's behavior, children must understand in general that people use symbolic language to communicate with computers, and they must select specific instructions to accurately represent their intended outcome for the robot's behavior.

Correspondence was first introduced in Lesson 3 of the curriculum, when students choose and sequence programming instructions to make a robot dance the Hokey-Pokey. Accomplishing this task requires children to identify the corresponding programming instruction for each line of the "Robot Hokey-Pokey" verse/dance. For example, a child who understands the correspondence between actions and instructions would find the programming instruction block with the "shake" symbol to recreate the line in which the robot "shakes it all about". To measure correspondence, children were assessed on how many of the correct instructions they chose.

#### 2.3.3. Sequencing instructions

Sequencing is a component of planning, and involves putting objects or actions in the correct order (Zelazo, Carter, Reznick, & Frye, 1997). To create a successful program, children must use procedural thinking and plan their programs in terms of a sequence of what happens *next*, *before*, or *until* another action (Pea & Kurland, 1984). In both literacy and mathematics, sequencing is essential: for putting phonemes, letters, words, or elements of a formula in the appropriate order (Neuman & Dickinson, 2002). Prior research with the TangibleK project showed that children who participated in the program earned significantly higher scores on a test of story sequencing than children who did not (Kazakoff & Bers, 2012; Kazakoff, Sullivan, & Bers, 2012).

In this curriculum, children were first introduced to the idea of sequencing instructions in Lesson 3's "Hokey-Pokey" challenge (described above). Sequencing was also a core component of Lessons 4–6, in which children had to properly arrange action instructions and increasingly complex control flow instructions in the correct order to achieve particular outcomes in the robot's behavior.

#### 2.3.4. Use of control flow instructions

"Control flow" refers to the concept that programmers can control the order in which a robot follows the instructions in its program through various programmatic methods. This curriculum introduced children to *control flow* instructions and *parameters*. Control flow instructions allow the robot to carry out instructions non-sequentially, e.g., in a loop, or only under certain conditions. For example, a CHERP program can include a "Repeat" control flow instruction in the following way: "Begin, Forward, Repeat 3 Shake, End-Repeat, Sing, End", to make the robot shake three times and then sing once. With the attachment of a light or touch sensor to the robot, *sensor parameters* can also be used to qualify the control flow instructions based on environmental stimuli. For instance, a child can program a robot to carry out an action or set of actions only "If (the environment is) Dark" or "If Light", and another set of actions "If Not (Light/Dark)". While there are currently no curriculum frameworks explicitly addressing control flow, these activities connect to mathematics, by reinforcing number sense and estimation, or to natural science, by comparing human and animal sensory functions with robot sensors. Children are also able to compare and contrast repeating or looping programs to patterns, cyclical events in the natural world, and calendar time. Children were assessed on their correct use of control flow structures in Lessons 4–6 and the final project.

## 3. Results

This section presents and compares children's achievement on programming and debugging concepts and other skills taught using the TangibleK robotics curriculum. Since the focus of this work is on computational thinking in a robotic context, the assessments presented here evaluate programming concepts instead of robotics knowledge. Children's work in each introductory lesson was assessed for two relevant programming concepts. These concepts, seven in total, were reassessed in the final project. Additionally, four debugging skills were assessed in all lessons and the final project. Each measure uses the Likert scale shown above, which ranges from 0 (did not *attempt* the task) or 1 (did not *complete* the goal, task, or understanding) to 5 (completely achieved the goal, task, or understanding). Analysis was conducted by aggregating scores from all classrooms and using paired-sample *t*-tests to compare scores on each concept from one lesson to the next. Findings are grouped by the powerful idea to which they relate. Note that the teacher in Classroom 1 chose not to formally teach Activity 6, so data for that Activity's items come only from two classrooms. A discussion about this choice is provided later in the paper.

### 3.1. Debugging

Average scores on the various debugging measures fell in the range of partial to mostly complete understanding and application of the skill (see Table 1). There was little variation in debugging scores between consecutive activities (see Fig. 1), with the exception that the average score on keeping the original goal was higher in Activity 4 than in Activity 5 (marked in Table 1). In other words, children's ability to keep working on the original goal (or choose an acceptable alternative) was higher in activities that did not require the use of sensors and sensor parameters. Scores on the other three components of debugging remained steady in the mid-to-upper range of the achievement scale across lessons.

**Table 1**
Student scores on debugging.

| | Debugging step 1 | | | Debugging step 2 | | | Debugging step 3 | | | Debugging step 4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *N* | Mean | SD | *N* | Mean | SD | *N* | Mean | SD | *N* | Mean | SD |
| Activity 1 | 43 | 4.19 | 0.98 | 45 | 4.22 | 0.80 | 42 | 4.00 | 1.08 | 43 | 4.16 | 1.05 |
| Activity 2 | 48 | 4.35 | 0.76 | 49 | 4.16 | 0.87 | 46 | 3.76 | 0.97 | 46 | 3.91 | 1.00 |
| Activity 3 | 31 | 3.97 | 1.11 | 39 | 4.21 | 0.95 | 33 | 3.67 | 1.27 | 34 | 3.77 | 1.28 |
| Activity 4 | 43 | 3.93 | 1.26 | 44 | 3.87* | 1.23 | 43 | 3.46 | 1.32 | 43 | 3.72 | 1.30 |
| Activity 5 | 30 | 3.53 | 1.17 | 36 | 3.50* | 1.23 | 27 | 3.22 | 1.19 | 30 | 3.50* | 1.23 |
| Activity 6 | 27 | 3.44 | 1.25 | 28 | 3.89 | 1.23 | 26 | 2.96 | 1.28 | 27 | 3.41 | 1.42 |
| Project | 28 | 3.97 | 0.92 | 29 | 4.42* | 0.64 | 27 | 3.93 | 0.92 | 26 | 4.08* | 0.69 |

*Note*. Classroom 1 did not do Activity 6.
*Denotes significant differences in the mean scores of the paired items at the $p < 0.05$ level. For Debugging Step 2 between Activities 4 and 5, $t(28) = 2.04$ and $p = 0.05$. For Debugging Step 2 between Activity 5 (the last activity completed by all classes) and the project, $t(19) = 3.12$ and $p = 0.01$. For Debugging Step 4 from Activity 5 to the project, $t(15) = 2.55$ and $p = 0.02$.

Repeated measures ANOVA analyses (see Table 2) were run for the four debugging skill variables across activities. The analyses were run across all seven activities (or in the case of Debugging Skill 1, across the four activities where this skill was assessed). In addition, a separate repeated measures ANOVA was run for each debugging skill variable for just Activities 1–5 since once classroom did not participate in Lesson 6 and the project lesson was unstructured.

Average debugging score did not vary significantly across activities when all lessons were considered. However, when removing the challenge activity and Lesson 6 where one class did not participate, a repeated measures ANOVA for each debugging variable did show a change across time, meaning there was, perhaps variation in debugging score across the more structure lessons, but, this variation averaged out when children worked on their own projects.

### 3.2. Powerful ideas of programming

In Activities 3 through 6 and in the culminating project, students completed specific programming challenges and were assessed on their ability to select instructions and put them in the order that would result in the goal behavior for the robot. Activities 4–6 also used special "control flow" instructions, which can tell the robot to loop through a set of actions repeatedly or to follow one "branch" of instructions or another based on sensor data.

#### 3.2.1. Choosing the correct programming instructions

The overall mean score on students' abilities to choose the correct instructions started off high in Activity 3. Scores then dropped, on average, over Activities 4 through 6, and returned to starting levels in the project (see Table 3 for detailed means). As mean scores fell to statistically significantly lower levels in Activity 4 and again in Activity 5, the percent of students reaching the target level of achievement also dropped. Seventy-six percent of students achieved in the target range on choosing programming instructions in Activity 3, which used only action instructions. In Activity 4, which introduced the first of the control flow instructions, 70% of children achieved the target level, as did only 46% in Activity 5, which added the use of sensors and sensor parameters, and 62% in Activity 6, which used a second, more challenging type of control flow instruction. However, 77% of children reached the target level of achievement on their projects – a similar rate to that in Lesson 3, the first activity to require choosing programming instructions (see Fig. 2).

#### 3.2.2. Control flow by sequencing

Sequencing ability was also introduced in Activity 3, along with making correspondences between intended robotic actions and programming instructions, when children made their robots dance the "Hokey-Pokey". Three-quarters of all students achieved in the target range in this first programming activity. Sequencing was also a core component of Activities 4–6, in which children had to properly arrange

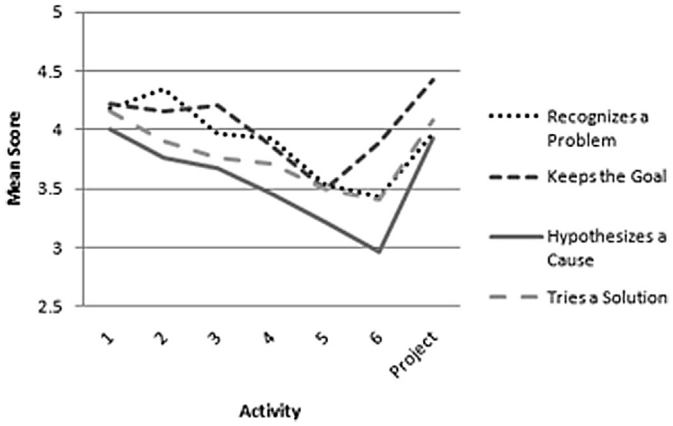


**Fig. 1.** Mean achievement on debugging across Activities. Average debugging scores for each activity and project. All four debugging components appear to follow a similar trend, but only the scores for keeping the original goal and attempting to solve the problem had statistically significant changes.

**Table 2**
Repeated measures ANOVAs by debugging steps.

| A repeated measures ANOVA was conducted to see if there was a significant difference in means of debugging level over time. | | | | | | |
|---|---|---|---|---|---|---|
| | All possible activities | | | Without Lesson 6 or project | | |
| | df | *F* | *p* | df | *F* | *p* |
| *Debugging Step 1* | | | | | | |
| Sphericity Assumed | (4, 16) | 1.559 | 0.23 | (2, 50) | 5.122 | 0.01** |
| *Debugging Step 2* | | | | | | |
| Greenhouse-Geisser Correction | (2, 4) | 5.740 | 0.07 | (3, 68) | 5.157 | 0.00** |
| *Debugging Step 3* | | | | | | |
| Greenhouse-Geisser Correction | (2, 4) | 4.586 | 0.10 | | | |
| Sphericity Assumed | | | | (4, 84) | 9.192 | 0.00** |
| *Debugging Step 4* | | | | | | |
| Greenhouse-Geisser Correction | (1, 3) | 2.682 | 0.23 | | | |
| Sphericity Assumed | | | | (4, 88) | 6.404 | 0.00** |

*Notes.* Classroom 1 did not do Activity 6. The Project was open-ended (children could choose to use less difficult blocks).
**Denotes significance at the $p < 0.01$ level.

both actions and increasingly complex control flow instructions in the correct order. In these activities, 59%, 53%, and 68% of children, respectively, achieved at the target level. Fewer children were able to reach the target level of achievement for sequencing in these activities than in Activity 3. A comparison of mean scores on sequencing from one activity to the next revealed a statistically significant drop between Activities 3 and 4, differentiating programs with actions only from those requiring two-part control instructions as well (see Table 4). As was seen with correspondence scores, the average sequencing score on children's projects was statistically significantly higher than the average score in Activity 5 (see Fig. 2).

#### 3.2.3. Control flow by special instructions

Activities 4–6 each introduced a new control flow instruction for creating looping or branching programs. Students, on average, achieved a "partially" complete understanding of the concepts (see Table 5 for detailed means). Less than 60% of students reached the target ("mostly" complete) level of understanding on all but one of these measures. (This degree of understanding was reached by 53% for looping, 60% for numeric parameters, 54% for sensor parameters, 68% for the first half of the conditional statement, and 41% for the second half of the conditional statement.)

There were no differences in average scores found between looping and conditional instructions or comparing the different types of parameters (see Fig. 3). The only statistically significant difference in scores was between the two parts of the conditional statement ("If" versus "If Not"). That is, children were, on average, more comfortable making the programming equivalent of the statement "If it's dark out, turn the light on" and less comfortable appending "If not, turn the light off" to that first statement.

### 3.3. Comparison of concepts between activities and projects

Differences in children's achievement on each of the above concepts from the introductory activities to the culminating projects were examined in two ways. First, children's scores from Activity 5 (the last activity completed by all classes) were compared to children's scores on the final project. This continued the comparison of scores on consecutive activities. Secondly, scores from the first activity that introduced a particular concept were compared to corresponding scores from the final project. For example, sequencing scores from Activity 3 (the first activity using that concept) were compared to sequencing scores on the final project. This comparison was done to address how children's scores on the same concepts might change with time and exposure. We should note that due to the self-selected nature of the final projects, not all children employed every concept to complete them, so *n* is relatively lower on these comparisons.

There were some statistically significant increases in scores from the final introductory activity completed by all classrooms to the culminating projects were seen on two overarching programming concepts: choosing the correct instructions (see Table 3) and sequencing the instructions to accomplish the goal (see Table 4), as well as on two elements of debugging (see Table 1): sticking with the original goal or choosing an acceptable alternative and taking steps to attempt to solve the problem. In fact, after these scores had dropped over the course of the activities, they returned to starting levels in the final projects (as described in the relevant sections above).

**Table 3**
Student scores on selecting programming instructions.

| | Selecting instructions | | | Comparison to subsequent activity | | | |
|---|---|---|---|---|---|---|---|
| | *N* | Mean | SD | Activity | df | *t* | *p* |
| Activity 3 | 45 | 4.24 | 1.07 | – | – | – | – |
| Activity 4 | 50 | 3.88 | 1.04 | 3 | 42 | 2.47 | .02* |
| Activity 5 | 41 | 3.34 | 1.20 | 4 | 38 | 2.77 | .01** |
| Activity 6 | 34 | 3.65 | 1.30 | 5 | 25 | −0.73 | 0.47 |
| Project | 48 | 4.15 | 0.92 | 6 | 29 | −1.19 | 0.25 |
| | | | | 5[a] | 35 | −3.12 | .00** |

*Note.* Classroom 1 did not do Activity 6.
**Denotes significance at the $p < 0.01$ level; *Denotes significance at the $p < 0.05$ level.
[a] This comparison was made as an alternative to the Activity 6-to-Project comparison as it was the last activity completed by all three classrooms prior to the project.

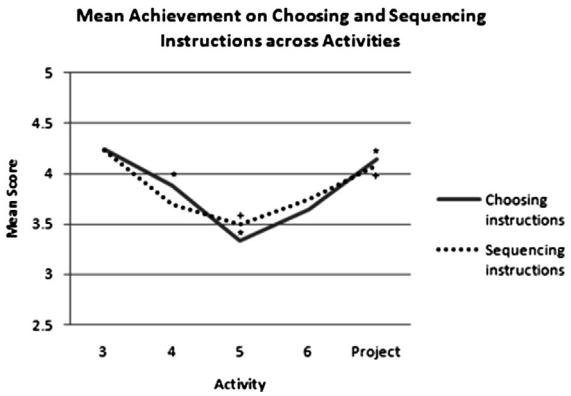


**Fig. 2.** Mean achievement on choosing and sequencing instructions average scores for choosing and sequencing programming instructions according to the goal. The dip on both choosing instructions (marked with *) and sequencing them (marked with +) at Activity 5 represents significantly lower scores as compared to Activity 4 and the Project. The drop in score for choosing instructions at Activity 4 is also statistically significant.

It was anticipated that the children's scores on the same concepts might increase with exposure, so comparisons were made between children's score on a concept in the activity that introduced it and the score on that same concept in the final project. However, there were no statistically significant differences seen in any such comparisons (see Table 6).

In summary, many children in each class reached the target level of achievement on the programming tasks over the course of the curriculum's six activities and culminating project. In the first three activities, which introduced the engineering design process, robotics, and programming, children's levels of achievement were particularly high (75%, on average, reaching target level of achievement). In Activities 4–6, which introduced more sophisticated concepts and programming instructions, fewer children (56%, on average) attained the same level of understanding.

Many children achieved high scores on properly selecting and sequencing instructions when the programming activities involved only action instructions ( ~75% for both skills) and in the final projects. Achievement was comparatively lower in activities which involved the conceptually and functionally more complicated control flow instructions and/or sensors (59% for both skills). Programs that use special control flow instructions visually appear linear, but the robot does not carry out one action per programming block, as it does with a program containing only actions instructions; the logical flow of the program may be a loop or forked path rather than a line. This introduces a conceptual complexity to programming with control flow instructions that does not exist with action instructions alone. Similarly, it appears based on relative scores that using the "If" instruction was simpler than using the "If Not" instruction (68% versus 41% target achievement). The complexity of each programming concept appears to be reflected in the portion of students who reached target levels of understanding.

## 4. Discussion

The results provide critical information on the accessibility of selected concepts from the fields of robotics and computer science for kindergarten children, adding clarity to developmentally appropriate learning expectations in order to revise and improve both the curricular activities and design features for early childhood robotics and programming technologies. The results also shed light on some of the challenges of conducting design-based research in a classroom setting.

One interesting feature of the results is the trend of decreasing achievement scores across Lessons 3–6. This is possibly related to the amount of time spent on each topic. Each activity in the curriculum introduced a progressively more challenging concept than the activity before it. In the later lessons, children were asked to build on concepts they had only recently learned. While each lesson was carefully designed to teach a particular topic and provide a space for exploration of it, these concepts may not have been fully ingrained or mastered yet while new material was introduced. This could also help explain lower scores in the later lessons.

Another interesting result relates to the several concepts for which children's average achievement scores increased from the final introductory lesson to the culminating project. With statistical significance, children averaged higher scores on choosing and sequencing

**Table 4**
Student scores on using sequencing for control flow.

| | Sequencing instructions | | | Comparison to subsequent activity | | | |
|---|---|---|---|---|---|---|---|
| | *N* | Mean | SD | Activity | df | *t* | *p* |
| Activity 3 | 49 | 4.23 | 1.16 | – | – | – | – |
| Activity 4 | 40 | 3.69 | 1.14 | 3 | 41 | 2.79 | .01** |
| Activity 5 | 40 | 3.50 | 1.20 | 4 | 38 | 1.07 | 0.30 |
| Activity 6 | 34 | 3.74 | 1.19 | 5 | 25 | 0.53 | 0.60 |
| Project | 49 | 4.08 | 1.04 | 6 | 30 | 0.25 | 0.80 |
| | | | | 5[a] | 35 | 2.14 | .04* |

*Note*. Classroom 1 did not do Activity 6.
**Denotes significance at the $p < 0.01$ level; *Denotes significance at the $p < 0.05$ level.
[a] This comparison was made as an alternative to the Activity 6-to-Project comparison as it was the last activity completed by all three classrooms prior to the project.

**Table 5**
Student scores on using special instructions for control flow.

| | Control flow instructions | | | Comparison to analogous concept | | | |
|---|---|---|---|---|---|---|---|
| | *N* | Mean | SD | Concept | df | *t* | *p* |
| Looping instruction | 49 | 3.61 | 1.17 | – | – | – | – |
| Numeric parameters | 50 | 3.76 | 1.17 | – | – | – | – |
| Sensor parameters | 41 | 3.59 | 1.12 | Numeric parameters | 38 | 0.87 | 0.42 |
| Conditional (If) | 34 | 3.74 | 1.24 | Looping instruction | 31 | 0.00 | 1.00 |
| Conditional (If Not) | 22 | 3.32 | 1.21 | Conditional (If) | 20 | 2.17 | .04* |
| | | | | Looping instruction | 19 | 0.65 | 0.53 |

*Note*. Classroom 1 did not use conditional statements.
*Denotes a statistically significant difference at the $p < 0.05$ level.

instructions during the final project than in any introductory activity except the first (and simplest) programming activity. While assistant from adults remained stable throughout all aspects of the curriculum and final projects, some other circumstances were different in the project compared to the lessons. The improved scores might be attributed to the fact that children had more enthusiasm for these personally-selected projects that would soon be part of a show-and-tell celebration as well as more time to experiment at their own pace than in the lessons. Alternatively, assuming children chose projects well-matched to their level of expertise, it would be reasonable to expect higher demonstrated levels of achievement as their projects likely focused on concepts children already felt more comfortable using. However, there were no statistically significant differences seen in comparisons of control flow instruction and sensor-related measures between the activity that introduced each concept and the culminating project. It is possible that even more time exploring these concepts was needed for significant learning gains to occur.

Surprisingly, children did not always perform better on simpler concepts than on more complex ones. For example, the lack of statistically significant differences between children's understanding of looping versus conditional programs and between numeric versus sensor parameters is unexpected both theoretically and based on anecdotal observations of these activities by researchers present during the activities. The concepts associated with looping and numeric parameters should, in principle, be more straightforward than those involved in programming with conditional statements and sensor parameters. Thus, at least somewhat higher levels of achievement on looping and numeric parameters had been expected compared to conditional statements and sensor parameters.

In some of the comparisons described above, the low *n* (less than half the overall study sample size) may have impacted the results. The statistically significant findings may have varied if, for instance, the students for whom researchers could not collect data tended to have above- or below-average achievement levels. As the activities in the curriculum increased in difficulty, the research assistants tended to provide increased support for children with questions, leaving less time to equally observe and assess all children. In fact, it was also observed that some children who perceived an activity to be difficult refrained from attempting it, resulting in no achievement scores for that activity and a lower *n* on those measures.

### *4.1. Curriculum discussion*

While it is beyond the scope of this paper to fully evaluate the TangibleK robotics curriculum, results indicate that the curriculum was generally engaging and developmentally appropriate for kindergarten students. Results point to kindergarten teachers being able to effectively implement the curriculum and to kindergartners being both interested in and able to learn and apply many aspects of robotics, programming, and computational thinking. However, the fact that fewer children achieved the target level of understanding on more complex topics than on the introductory concepts might indicate that the curriculum should devote more time for children to build up to and fully explore the complex material in order to fully understand it. In order to test this, a new iteration of the curriculum is currently

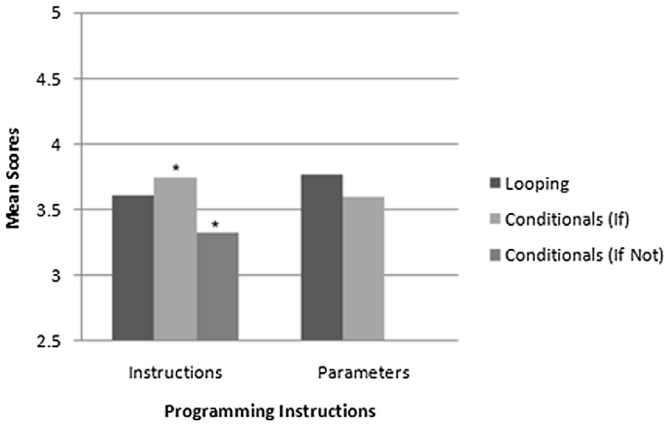


**Fig. 3.** Mean achievement on control flow concepts. A comparison of average scores for the different types of control flow instructions and parameters. The only significant difference in scores was between the two conditional instructions (marked with *).

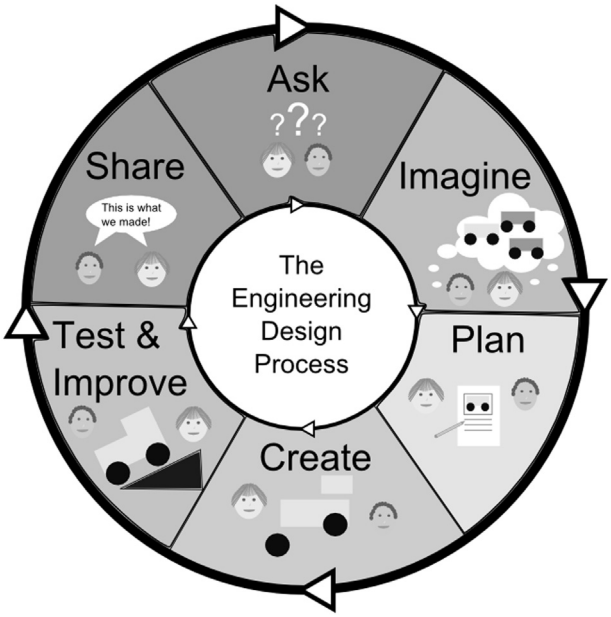


**Fig. 4.** An illustration of the engineering design process.

being developed that will divide the prior curriculum based on action and sensing, and expand the number of lessons and amount of time spent exploring each topic (particularly the more complex ones) in both structured and free-play-based formats to provide further opportunities for students' investigation of concepts and to reinforce their learning. Additional supporting activities will also be added. Kindergarteners vary widely in their levels of cognitive development and learning abilities, and such adaptations to the TangibleK curriculum may address this range even more than the current format already does.

The results also demonstrate the complexity of assessing sophisticated learning processes in a classroom setting. There was a necessary trade-off built into the study design: gathering an adequately detailed picture of children's learning had to be balanced with keeping data collection feasible given that each adult was working with several children in the context of a full classroom. In some cases (particularly the later activities), a different setting, such as individual-child sessions, may have provided a better context for some students to demonstrate their abilities. However, the goal of the study was to examine the TangibleK program in a typical kindergarten classroom, and this endeavor was successful. Although ultimately some data could not be collected from every student on every measure, information was gathered about the reality of implementing the curriculum in classroom settings and the supports necessary to meet the needs of all students.

### *4.2. Limitations of the study and future directions*

The TangibleK curriculum was taught during regular school hours in three schools in the Greater Boston area. There were both benefits and drawbacks to conducting research in a school setting rather than an experimental setting. By testing the curriculum as taught by kindergarten teachers, in both public and private schools, we have demonstrated that, given professional development in robotics education, a dedicated teacher can successfully teach this content in her or his own classroom. However, as with any study that takes place in a school setting, the present study faced several environmental limitations. While each of the participating teachers taught the same curriculum, it is impossible to control for all teacher, classroom, and school variations that may have influenced results. For example, the three teachers in this study were very different from one another. While some teachers allowed their class to work through difficult concepts on their own, others gave more step-by-step instructions. Teachers were given leeway to teach the curriculum in whatever way they believed best suited the needs of their classrooms, however, this causes methodological issues for data analysis. Further research should be conducted with a focus on how teaching styles and classroom culture serve to enhance or hinder a robotics curriculum.

**Table 6**
Students scores on concepts in culmination project.

| | Project scores | | | | Comparison to introductory activity | | | |
|---|---|---|---|---|---|---|---|---|
| | *N* | Mean | SD | % Scoring 4+ | Activity | df | *t* | *p* |
| Choosing instructions | 48 | 4.15 | 0.92 | 77.0 | 3 | 40 | 0.70 | 0.49 |
| Sequencing | 49 | 4.08 | 1.04 | 73.5 | 3 | 40 | 0.78 | 0.44 |
| Repeats | 40 | 3.75 | 1.26 | 62.5 | 4 | 36 | 0.77 | 0.45 |
| Numeric parameters | 32 | 3.94 | 1.22 | 65.6 | 4 | 30 | 0.77 | 0.45 |
| Sensors | 18 | 4.05 | 0.73 | 83.4 | 5 | 15 | 0.94 | 0.36 |
| Sensor parameters | 20 | 3.90 | 0.85 | 70.0 | 5 | 17 | 0.36 | 0.73 |
| Ifs[a] | 14 | 3.57 | 0.80 | 78.6 | 6 (If) | 13 | 0.27 | 0.79 |

*Note*. Classroom 1 did not do Activity 6.
[a] There was no separate measure for using "If Not" instructions in the projects.

Another drawback the study encountered was a fluctuating number of daily participants. Children were fairly regularly absent, temporarily out of the classroom, or otherwise unable to participate in the class. Other times, the busy classroom and divided adult attention prevented assessments from being collected for all children, particularly if an assessment required long and sustained periods of observation. Teacher differences also impacted the low number of participants in some activities. For example, one teacher chose not to teach Lesson 6 in order to have more time to review previous concepts before the final project, drastically lowering the *n* for that lesson. Further research should be done expanding the scope of this study by gathering more participants and, if possible, ensuring more consistent completion of each activity.

The present study inspires additional research agendas. While the focus of this work is on kindergartners, further investigations should look at the way younger (PreK-K) and older (1st–2nd grade) students are able to learn and apply the same powerful ideas. It would be important to determine whether some of the concepts that were particularly challenging to the kindergartners in this study pose less of a challenge with longer exposure or if introduced when children are older. Further research will also expand the overall sample size as well as the age and experience range of the sample. Other work should attempt to assess the feasibility of implementing this curriculum for a classroom teacher with typical support staff, that is, with minimal involvement of research assistants except for training teachers and conducting data collection. In the present research, participating teachers each had about three trained assistants in the classroom to help troubleshoot technology issues, assess the children's progress, and provide one-on-one help as needed. For this curriculum to become widespread, it will be necessary to know more about what supports teachers need (modifications to the curriculum, classroom management alternatives, additional adult support, etc.) to successfully implement the curriculum. Finally, it is beyond the scope of this current study, but a follow-up study could look at longitudinal or transfer effects of the TangibleK curriculum. What concepts do the students retain? How is computational thinking having an impact in other areas of their academic and extra-curricular lives? Are children able to apply the engineering design process to other subject areas after completing this curriculum? Further research should look at the long-term benefits of incorporating programming and robotics into early childhood education.

It is important to note that many of the challenges that arose as part of the present study were posed by the robotics hardware itself, and not the curricular activities. This highlights the importance of making developmentally appropriate hardware and software specifically designed for young children. Results show that for the children in this study, correctly connecting robotic parts proved more challenging than understanding the function of each part or the underlying computational concept. This result is not surprising since the CHERP programming interface and the curricular activities introducing the robot's parts and their purposes were specifically developed for kindergartners as part of this research project, while the robotics kit hardware was designed for older children as part of a commercially available LEGO® product. Furthermore, children spent a significant amount of time fixing their robots, which came apart frequently. It was challenging for many children to assemble some of the pieces on their own, and they needed adult help. If children had spent the robot repair time working on their computer programs instead (and if teachers were able to spend that time providing support for learning the central concepts rather than helping re-build robots), perhaps children would have attained higher levels of achievement in their understanding of complex powerful ideas involved in computational thinking.

The findings from this study have informed the TangibleK Project, in which early childhood teachers (pre-kindergarten through 2nd grade) will systematically implement a robotics curriculum revised according to several of the points outlined above. The teachers will document their experiences and their students' learning outcomes over the course of a school year using KIWI, a developmentally appropriate robotics hardware that will replace the LEGO® hardware used in this current study.

Despite the limitations of the study described in this paper, post-study data collected from the teachers speaks to the success of the TangibleK Robotics program. All the teachers said they would participate in TangibleK again if given the chance. Along with the general success and enthusiasm of the children, this feedback highlights the overall positive and educational nature of the experience.

## 5. Conclusion

The early childhood classroom is not typically a place where we expect to find students programming robots. Yet, with the availability of developmentally appropriate technologies, this is increasingly possible, and the result may be the advancement of technological fluency in our nation's youth. This paper explored the TangibleK Robotics Program as a viable option for classroom teachers to integrate developmentally appropriate technology education into the early childhood classroom.

With CHERP, children spend their time building a robot, planning its actions, using physical wooden block or the computer screen to construct programs, and iteratively improving the robot and programs according to initial goals and subsequent discoveries. Because the tangible programs and robots exist off-screen, children are drawn to investigate the work of other children, work collaboratively, and negotiate sharing materials, as well as develop their fine-motor skills. These artifacts serve as points of discussion and reminders of the activity content even after the computer has been turned off. As the analysis presented in this paper has explored, in this rich process of creation in both the physical and digital worlds, children actively engage in problem-solving and learn powerful ideas from computer science and robotics, including core concepts of computational thinking.

Research is essential to understanding the impact of new technologies on the development of children and how children are using and could be using these tools. As parents, educators, policymakers, and researchers it is our responsibility to ensure our children receive the technological education needed for healthy development and a successful future. The TangibleK Robotics Program introduced in this paper shows that when given age-appropriate technologies, curriculum and pedagogies, young children can actively engage in learning from computer programming as applied to the field of robotics. They can then take their first steps into developing computational thinking.

## Acknowledgments

The TangibleK project was supported by National Science Foundation (NSF) DRL-0735657. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

The authors would like to thank participating schools and teachers for their commitment to and participation in this project.

## References


Barron, B., Cayton-Hodges, G., Bofferding, L., Copple, C., Darling-Hammond, L., & Levine, M. (2011). *Take a giant step: A blueprint for teaching children in a digital age*. New York: The Joan Ganz Cooney Center at Sesame Workshop.

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. http://dx.doi.org/10.1145/1929887.1929905.

Bers, M. U. (2008). *Blocks, robots and computers: Learning about technology in early childhood*. New York: Teacher's College Press.

Bers, M. U. (2010). Beyond computer literacy: supporting youth's positive development through technology. *New Directions for Youth Development, 128*, 13–23.

Bers, M. U. (2012). *Designing digital experiences for positive youth development: From playpen to playground*. Oxford University 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: Childhood in a digital world* (pp. 49–70). Greenwich, CT: Information Age Publishing.

Bers, M. U., Ponte, I., Juelich, K., Viera, A., & Schenker, J. (2002). *Teachers as designers: Integrating robotics into early childhood education*. Information Technology in Childhood Education.

Cejka, E., Rogers, C., & Portsmore, M. (2006). Kindergarten robotics: using robotics to motivate math, science, and engineering literacy in elementary school. *International Journal of Engineering Education, 22*(4), 711–722.

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

Gelman, R., & Brenneman, K. (2004). Science learning pathways for young children. *Early Childhood Research Quarterly (Special Issue on Early Learning in Math and Science), 19*(1), 150–158.

Horn, M. S., Davis, P., Hubbard, A., Keifert, D., Leong, Z. A., & Olson, I. C. (June 2011). Learning sustainability: children, learning, and the next generation eco-feedback technology. In *Proc. 10th international conference on interaction design and children (short paper)*. Ann Arbor, MI.

International Society for Technology in Education. (2007). *NETS for students 2007 profiles*. Washington, DC: ISTE. Retrieved from www.iste.org/standards/nets-for-students/nets-for-students-2007-profiles.aspx#PK-2.

International Society for Technology in Education and The Computer Science Teachers Association. (2011). *Operational definition of computational thinking for K-12 thinking-operational-definition-flyer.pdf*. International Society for Technology in Education and The Computer Science Teachers Association.

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.

Kazakoff, E., Sullivan, A., & Bers, M. (2012). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal, 41*(4), 245–255.

Klahr, D., & Carver, S. (1988). Cognitive objectives in a LOGO debugging curriculum: instruction, learning, and transfer. *Cognitive Psychology, 20*, 362–404.

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.

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

NAEYC & Fred Rogers Center for Early Learning and Children's Media. (2012). *Technology and interactive media as tools in early childhood programs serving children from birth through age 8. Joint position statement*. Washington, DC: NAEYC. Latrobe, PA: Fred Rogers Center for Early Learning at Saint Vincent College. Retrieved from www.naeyc.org/files/naeyc/file/positions/PS_technology_WEB2.pdf.

Neuman, S. B., & Dickinson, D. K. (Eds.). (2002). *Handbook of early literacy research*. New York: Guilford Press.

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

Papert, S. (2000). What's the big idea? Toward a pedagogy of idea power. *IBM Systems Journal, 39*(3 & 4), 720–729. http://dx.doi.org/10.1147/sj.393.0720.

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

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

Piaget, J. (1954). *The construction of reality in the child*. New York: Basic Books.

Resnick, M. (2003). Playful learning and creative societies. *Education Update, 8*(6). Retrieved from http://web.media.mit.edu/~mres/papers/education-update.pdf.

Rogoff, B., Goodman Turkanis, C., & Bartlett, L. (2001). *Learning together: Children and adults in a school community*. New York, NY: Oxford University Press.

Salomon, G., & Perkins, D. N. (1987). Transfer of cognitive skills from programming: when and how? *Journal of Educational Computing Research, 3*, 149–169.

Sesame Workshop. (2009). Sesame workshop and the PNC Foundation join White House effort on STEM education. Retrieved from http://www.sesameworkshop.org/newsandevents/pressreleases/stemeducation_11212009.

Stair, R. M., & Reynolds, G. W. (2003). *Principles of information systems* (6th ed.). Boston, MA: Course Technology – ITP.

U.S. Department of Education, Office of Educational Technology. (2010). *Transforming American education: Learning powered by technology*. Washington, DC: U.S. Department of Education, Office of Educational Technology. Retrieved from http://www.ed.gov/technology/netp-2010.

White House. (2011). Educate to innovate. Retrieved from http://www.whitehouse.gov/issues/education/educate-innovate.

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

Zelazo, P. D., Carter, A., Reznick, J. S., & Frye, D. (1997). Early development of executive function: a problem-solving framework. *Review of General Psychology, 1*(2), 198–226.

Zigler, E. F., & Bishop-Josef, S. J. (2006). The cognitive child vs. the whole child: lessons form 40 years of Head Start. In D. G. Singer, R. M. Golinkoff, & K. Hirsh-Pasek (Eds.), *Play = learning: How play motivates and enhances children's cognitive and social-emotional growth* (pp. 15–35). New York, NY: Oxford University Press.