Journal of Information Technology Education: Innovations in Practice

An Official Publication of the Informing Science Institute
InformingScience.org

JITEiip.org

Volume 16, 2017

THE IMPACT OF USER INTERFACE ON YOUNG CHILDREN'S COMPUTATIONAL THINKING

Alex Pugnali* Tufts University, Medford, MA, USA Alex.Pugnali@tufts.edu
Amanda Sullivan Tufts University, Medford, MA, USA Amanda.Sullivan@tufts.edu
Marina Umaschi Bers Tufts University, Medford, MA, USA Marina.Bers@tufts.edu
* Corresponding author

ABSTRACT

Aim/Purpose	Over the past few years, new approaches to introducing young children to computational thinking have grown in popularity. This paper examines the role that user interfaces have on children's mastery of computational thinking concepts and positive interpersonal behaviors.
Background	There is a growing pressure to begin teaching computational thinking at a young age. This study explores the affordances of two very different programming interfaces for teaching computational thinking: a graphical coding application on the iPad (ScratchJr) and tangible programmable robotics kit (KIBO).
Methodology	This study used a mixed-method approach to explore the learning experiences that young children have with tangible and graphical coding interfaces. A sample of children ages four to seven ($N = 28$) participated.
Findings	Results suggest that type of user interface does have an impact on children's learning, but is only one of many factors that affect positive academic and socio-emotional experiences. Tangible and graphical interfaces each have qualities that foster different types of learning
Keywords	robotics, coding, early childhood, user interfaces, collaboration, computational thinking

INTRODUCTION

New technologies for learning and playing are growing in prevalence amongst young children under the age of eight. A recent study by Common Sense Media found that two-thirds of children under the age of eight have access to a console video game player at home, and 35% have access to a

Accepted by Editor Keith Willoughby | Received: January 17, 2017 | Revised: May 8, 2017 | Accepted: June 15, 2017.
Cite as: Pugnali, A., Sullivan, A., & Bers, M. U. (2017). The impact of user interface on young children's computational thinking. *Journal of Information Technology Education: Innovations in Practice, 16,* 171-193. Retrieved from http://www.informingscience.org/Publications/3768

handheld game player such as a Game Boy, PlayStation Portable (PSP), or Nintendo DS. Additionally, there has been a five-fold increase in ownership of tablet devices such as iPads, from 8% of all families in 2011 to 40% in 2013 (Common Sense Media, 2013). In addition to these screen-based technologies, new tangible technologies, such as robotics kits, have also been growing in popularity with young children during the past few years. Prior research has shown that children as young as four years old can build and program a simple robot (Bers, 2008; Bers, Ponte, Juelich, Viera, & Shenker, 2002; Cejka, Rogers, & Portsmore, 2006; Kazakoff, Sullivan, & Bers, 2013; Perlman, 1976; Sullivan & Bers, 2015; Wyeth, 2008).

With this increase in popularity of technological tools and toys, federal education programs and private initiatives in the United States have focused on improving technological literacy and making computational thinking a priority in early childhood school settings (e.g. U.S. Department of Education, 2010). *Computational thinking* involves a set of skills that include problem-solving, design, and systematic analysis (Bers, 2010; Bers, 2017). Computational thinking represents a type of analytical thinking that shares many similarities with mathematical thinking (e.g., problem solving), engineering thinking (designing and evaluating processes), and scientific thinking (systematic analysis) (Bers, 2010; Bers, 2017). While the act of engaging in computational thinking is rooted in computer science, some have argued that it is a skill that is fundamental for everyone to master, just like reading, writing, and arithmetic (Wing, 2006). In a pivotal article on the need to expand the reach of computational thinking, Jeanette Wing (2006) states that it represents a universally applicable attitude and skill set for everyone, not just for computer scientists.

There is a growing pressure to teach computational thinking beginning in early elementary school and this has put teachers, administrators, and parents in a difficult position when investing in technology for early childhood education as they are faced with an ever-changing array of digital tools now being marketed to this age group. They are faced with questions such as, "what computational thinking skills are my children actually learning with this tool?" or "will my child find this tool fun and engaging?" "How can I successfully integrate it with my curriculum?" When choosing tools for school or home, parents and teachers must now also navigate the choice between screen-based and non-screen-based technologies that are quickly spreading out.

The pilot study described here provides data to answer these questions by exploring affordances of two very different programming interfaces: a graphical coding application on the iPad (ScratchJr) and tangible programmable robotics kit (KIBO). Both programming languages claim to teach many of the same introductory computational thinking skills to young children, but through very different interfaces. The goal of this work is to understand if these two interfaces offer different learning experiences for young children when it comes to computational thinking. In addition to examining children's learning, this study examines whether the type of technological interface (tangible versus graphical) impacts children's positive behaviors and interactions. This is crucial when thinking about tools for the early childhood classroom. This is explored through the context of Bers' (2012) Positive Technological Development (PTD) framework. PTD is an extension of the computer literacy and the technological fluency movements but adds psychosocial and ethical components to the cognitive ones and focuses on using technology to promote student engagement and collaboration. Results from this study are also analyzed through this PTD lens. Finally, this paper presents implications for choosing developmentally appropriate technology to meet both the learning and socio-emotional goals of students and teachers.

LITERATURE REVIEW

COMPUTATIONAL THINKING

The term "Computational Thinking" has been defined in many ways and encompasses a broad 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 Sci-

ence Teachers Association, 2011; Lee et al., 2011). According to a framework by Brennan & Resnick (2012), computational thinking involves the *concepts* designers engage with as they program, *practices* designers develop as they engage with the concepts, and *perspectives* designers form about the world around them and about themselves. These concepts may include very specific programming concepts (such as, repeat loops or sequencing), the practices may include methods of problem-solving or collaboration, and perspectives may include questioning things beyond the interface that is being worked with (such as, how are other things in the world automated?).

Approximately in 2010, the issue of computational thinking in K-12 education took center-stage following a stark report by Wilson, Sudol, Stephenson, and Stehlik (2010) that revealed very low numbers for women in computing and that more than two-thirds of the country had few computer science standards at the secondary school level (Grover & Pea, 2013). Since then, public and private organizations began to focus on programs, frameworks, and initiatives to foster computational thinking and address these issues. For example, that same year (in 2010), the International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) led a National Science Foundation project entitled, "Leveraging Thought Leadership for Computational Thinking in PK-12".

Fostering computational thinking through learning to code was brought to the national spotlight in the United States in 2014 when President Barack Obama wrote his highly publicized first line of JavaScript and became one of over 100 million people worldwide to have participated in Code.org's Hour of Code event. Prior to that, in 2013, New York City Mayor Bill de Blasio launched the Tech Talent Pipeline, aiming to give hundreds of after-school programs access to free computer science learning materials from Google. Most recently, in 2016, a major collaboration between the Association for Computing Machinery, Code.org, the Computer Science Teachers Association, the Cyber Innovation Center and the National Math and Science Initiative have collaborated with states, districts, and the computer science education community to develop conceptual guidelines for computer science education (K12 Computer Science Framework, 2016).

It is important to note that the benefits of learning computational thinking skills are not limited to the realm of technological literacy. It is a type of analytical thinking that shares many similarities with mathematical thinking (e.g., problem solving), engineering thinking (designing and evaluating processes), and scientific thinking (systematic analysis) (Bers, 2010). Children as young as four years old can learn foundational computational thinking concepts (Bers, 2008; Bers, 2012) and this kind of learning can support their literacy, mathematical, and socio-emotional development (Kazakoff & Bers, 2012; Kazakoff et al., 2013). Teaching computational thinking also allows students to practice problem-solving skills such as trial and error (Barr & Stephenson, 2011). The issue presently facing the field of educational technology is not whether to teach computational thinking in early childhood, but how to best select developmentally appropriate tools and curricula to do so.

Tangible versus Graphical Interfaces

In order to teach foundational computational thinking skills to young children, a new range of programming and robotics applications have emerged over the past few years. Young children beginning in pre-kindergarten can use simple programming interfaces to create interactive robotics projects (Bers et al., 2002; Cejka et al., 2006; Elkin, Sullivan, & Bers, 2016; Perlman, 1976; Wyeth, 2008) and graphical animation projects (Portelance, Strawhacker, Bers, 2015; Strawhacker & Bers, 2015). These types of projects can help young learners engage with powerful ideas from technology, including many computational thinking concepts that can serve them in educational and personal pursuits throughout their lives (Bers, 2008).

Graphical user interfaces, in the form of programming applications on tablets and computers, have gained popularity in recent years, in part due to new federal and private initiatives making technological literacy a priority in schools (e.g. U.S. Department of Education, 2013). For example, the free programming language Scratch (www.scratch.mit.edu), which was designed with users ages 8-16 in

mind, has thrived in recent years with over 12,000,000 registered users (Resnick et al., 2009; https://scratch.mit.edu/statistics/). The Scratch programming language allows users to engage with foundational concepts such as sequencing, repeat loops, and variables to create games, stories, and videos through coding (Resnick et al., 2009).

While many graphical applications, like Scratch, focus on older children and adolescents, research shows that children as young as four years old can master fundamental programming concepts of sequencing, logical ordering, and cause-and-effect relationships (Bers, 2008; Fessakis, Gouldi & Mavroudi, 2013; Kazakoff & Bers, 2011). Since then, newer languages, like ScratchJr (one of the tools used in this study) and Daisy the Dinosaur, have been designed to introduce younger children ages five to seven to fundamental concepts of computer programming and computational thinking (Strawhacker, Lee, Caine, & Bers, 2015). These applications use colorful images and graphical programming blocks to engage young children in foundational computer programming concepts as they create on-screen animations.

Tangible technologies, like robotics such as KIBO (the other tool in this study), have grown in popularity in recent years as well, especially with recommendations from the American Academy of Pediatric for limited screen-time for young children (American Academy of Pediatrics, 2003). New robotics kits have evolved to become a modern generation of learning manipulatives that help children develop a stronger understanding of mathematical concepts such as number, size, and shape in much the same way that traditional materials like pattern blocks, beads, and balls once did (Bers, 2008; Brosterman 1997; Resnick et al. 1998). Tangible robotics kits may also open young children up to different kinds of learning as well. For example, tangible robotic manipulatives allow children to develop fine motor skills and hand-eye coordination while also engaging in collaboration and teamwork (Bers, Seddighin, & Sullivan, 2013; Lee, Sullivan, & Bers, 2013).

With the growing mix of both graphical (i.e. onscreen), tangible (physical, hands-on), and hybrid (combined graphical and tangible) interfaces becoming readily available to teach foundational computational thinking skills to young children, parents and educators must make careful choices about the learning affordances of both types of interfaces. There has been a growing focus on investigating in tangible interface learning, however there is still little empirical evidence that tangible interfaces offer educational benefits compared to graphical or other digital methods (Horn, Crouser, & Bers, 2012; Marshall, 2007). In research by Horn et al. (2012) on a museum exhibit using Tern, a tangible and graphical robotic programming language, tangible interfaces were shown to be more appealing to children (ages 16 and under) and more useful for fostering collaboration. These findings have been echoed by similar studies (Cheng, Der, Sidhu, & Omar, 2011; Manches & Price, 2011).

However, a more recent study by Strawhacker and Bers (2015) studying tangible and graphical programming interfaces with young children in a Kindergarten setting, the researchers found there was little association between user interface and programming comprehension, although there may be an order-effect when introducing user interfaces. This study by Strawhacker and Bers is one of the few existing studies comparing the learning impact of tangible versus graphical programming interfaces on young children. However, Strawhacker & Bers did not examine the impact of interface on student engagement, collaboration, or other positive behaviors that are a major component of early childhood development. The present study follows up on this research in order to examine the impact of interface on programming knowledge and computational thinking as well as student engagement. It focuses on two programming languages that are currently widely available: the ScratchJr programming language and the KIBO robotics kit.

METHOD

RESEARCH QUESTIONS

This pilot study described here used a mixed-method approach in order to explore the learning experiences that young children have with tangible and graphical coding interfaces and to measure the

computational thinking skills that children using these tools gain. The aim of the study is to answer the following questions:

- What impact do tangible vs graphical interfaces have on the children's **positive technological development**?
- What impact does the type of interface have on the **computational thinking skills** that children gain?

Participating children completed a weeklong robotics and programming summer program hosted by the Developmental Technologies (DevTech) Research Group at Tufts University. Upon completion of the week, children's mastery of computational thinking skills were measured.

Sample

The sample in this study came from a group of children ages 4-7 admitted into a summer program at DevTech. Students were recruited through advertisements sent to students who participated in previous summer programs, DevTech social media, schools previously worked with, and various networks at Tufts University. Children in the camp came from across the New England region. A requirement of the camp was that all students were rising kindergarten-second grade students. There was a total of 60 children across three sessions of the program.

Registration for the summer program was on a first-come, first-served basis, with an enrollment cap of 14 children per technology/session. There was no application involved and children were guaranteed a spot once registration was filled out and the $200 registration fee was submitted. There were no scholarships offered for those who were interested but could not afford the registration fee. Children could attend two weeks using different technologies, but not more than one week of the same technology (i.e. two weeks of ScratchJr). Two of the three weeks of KIBO sessions reached capacity one month prior to their start date, so there were some families who were interested but unable to register. No ScratchJr sessions reached capacity.

Of the 60 children in the camp, a total sample of N=28 participants were included in the research study. Participants were included based on parental consent and whether or not they had previous experience attending the summer program or previous experience using the technologies being taught (only those with no prior experience are included in this research). Of the total sample n=14 were in the tangible KIBO robotics group (Kindergarten=5, First=6, Second=3; Male = 12, Female = 2) and n=14 were in the graphical ScratchJr group (Kindergarten=6, First=4, Second=4; Male = 6, Female = 8). Participants in both the tangible and graphical had an average age of 5.86 years old.

Procedure

Children participated in an intensive week-long programming-based curricula using either the tangible KIBO robotics kit, or the graphical ScratchJr tablet application. Participating children signed up for the program of their choice and were not randomly assigned to the groups. When registering, parents had the option to enroll their children in either KIBO or ScratchJr. Due to the nature of each technology and parents' educational goals for their children, we decided to not randomly select the technology that each participant used. The KIBO Robotics Kit used in the summer program costs $350 while ScratchJr is a free tablet app. Some parents wanted their children exposed to a technology they could not otherwise afford, while others wanted us to give their child some skills in the program that they could continue exploring at home.

Each program lasted five days, with approximately three hours of curricular instruction each day. Each program had a head counselor that guided students through using the technology and guided both large group and individual activities. Counselors were supported by three research assistants whose roles included observing children during activities and conducting research assessments with children.

Counselors and research assistants received training on the technology they were implementing, the pre-written curriculum they were teaching, and research methods being implemented. Training lasted for three hours and was split into two parts. During the first half of the training session both counselors and research assistants were given an overview of how to use both technologies, the curricula being taught, and the research methods being implemented throughout the week-long session. For the second half, counselors were given time to go through and practice each activity they would be leading with their students and were given additional training on the technology they were using. Research assistants were given a more in-depth briefing of the research protocol and were instructed on how to administer the computational thinking assessments.

Children were a part of one of two conditions, based on the program they signed up for: tangible learning using the KIBO Robotics kit; or graphical learning using the ScratchJr tablet app. The curriculum for both technologies followed the same theme, "Going on a Safari" and explored the same computational thinking concepts: sequencing, repeat loops, and conditionals. The activities themselves were slightly different due to the nature of the technology being used. The following sections describe the KIBO and ScratchJr interfaces in more detail.

Tangible: KIBO Robotics Kit

This study uses the KIBO robotics kit to examine the impact of a tangible programming interface on children's mastery of computational thinking skills. KIBO is a programmable robotics kit specifically designed for young children aged 4-7 years old developed by Marina Umaschi Bers and Mitch Rosenberg. Using KIBO, children assemble their own mobile robot with motors, wheels, and sensors and program it to move the way they want with wooden programming blocks (see Figure 1).

KIBO was chosen to represent a tangible programming technology because it is programmed without any screen time from tablets or computers. Children create a sequence of instructions (i.e., a program) using interlocking wooden programming blocks that represent different actions for KIBO to carry out. KIBO uses an embedded scanner to read the barcode on each programming block, and the completed program can be run with the pressing of a button. Prior research has shown that children in pre-kindergarten through second grade can learn engineering and programming concepts with KIBO (Elkin et al., 2016; Sullivan, Elkin, & Bers, 2015).

Figure 1. The KIBO robot

KIBO's programming language is composed of over 18 individual wooden programming blocks (see Figure 2). Some of these blocks represent simple motions for the KIBO robot, such as move Forward, Backward, Spin, and Shake. Other blocks represent complex programming concepts such as Repeat Loops and Conditional "If" statements.

Figure 2. KIBO's programming language

Graphical: ScratchJr

For the graphical interface, this study used the ScratchJr application. ScratchJr is an introductory programming language for the iPad and Android tablet that enables young children (ages 5-7) to create their own interactive stories, collages, and games (Strawhacker et al., 2015). Using ScratchJr, children snap together graphical programming blocks to make characters move, jump, dance, and sing. Children can modify characters in the paint editor, add their own voices and sounds, as well as insert photos of themselves or other images taken using the tablet's camera. Prior research has shown that children in kindergarten through second grade can successfully learn foundational programming concepts with ScratchJr (Portelance & Bers, 2015; Portelance et al., 2015).

ScratchJr's programming language is organized into six categories of onscreen blocks. The categories are represented by different colors: yellow Trigger blocks, blue Motion blocks, purple Looks blocks, green Sound blocks, orange Control flow blocks, and red End blocks. The blue palette of programming instructions lies along the center of the editor. Children display one instruction category at a time by clicking selectors on the left. Dragging instruction blocks from the palette into the scripting area below activates them. Snapping blocks together creates programs that are read and played from left to right (see Figure 3).

Comparing KIBO and ScratchJr

Both KIBO and ScratchJr were uniquely developed to reach an audience of young children in Kindergarten through second grade. Both were developed by educational technology experts at the DevTech Research Group. In fact, the two programming languages share many similarities. For example, in both cases categories of programming blocks are color coded. In many instances, there are even overlaps in these categories (e.g., in both languages "blue" connotes motion blocks). Unlike other programming languages developed for older children which ask users to write programs from top to bottom, both KIBO and ScratchJr ask children to write code from left to right- just like they

are learning to read. Finally, both languages introduce children to core computational concepts of sequencing, repeat loops, and conditional statements.

Figure 3. ScratchJr interface

Despite the many similarities the languages share, they also differ in a few key ways. The biggest difference is also the most obvious one and the focus of this paper: one language is onscreen (ScratchJr) and one is not (KIBO). Being that KIBO is a tangible interface, the kit allows children to explore engineering concepts of building, designing, and constructing with motors and sensors while they are programming their robot. While ScratchJr does not allow for physical building, the onscreen interface makes it easier to create longer programs than with KIBO's tangible block language. Additionally, it allows children to create multiple programs for the same character, which you cannot do with KIBO. This paper explores whether these key differences between tangible and graphical programming languages for children results in different mastery of core computational concepts.

Curriculum

The curricula used during the week-long summer programs was the same across both sections. Each weeklong session included 15 hours of curricular instruction through a range of different activities, some of which used technologies and others that focused on art, music, and dance.

This study not only looks to examine computational thinking, but also the overall learning experience and engagement of students through the PTD framework (Bers, 2012). Research about children's engagement in learning settings describes both the psychological and behavioral characteristics of what it means to be "engaged" (Brewster & Fager 2000; Finn & Rock 1997; Marks 2000). Psychologically speaking, engaged students are intrinsically motivated by curiosity, interest, and enjoyment, and are likely to want to achieve their own intellectual or personal goals (Brewster & Fager 2000; Finn & Rock 1997; Jablon & Wilkinson, 2006; Marks 2000). Children who are highly engaged also demonstrate positive behaviors such as concentration, investment, enthusiasm, and effort (Brewster & Fager 2000; Finn & Rock 1997; Jablon & Wilkinson, 2006; Marks 2000).

Inspired by this work, the curriculum for the summer program was designed to promote positive behaviors that are indicative of student engagement. The Positive Technological Development (PTD) framework developed by Bers (2012) was used as a guiding framework in this study to define student engagement with technology. This provides a model for how children's personal and social

development can be fostered through the use of technology. PTD proposes six positive behaviors (the "six Cs") that should be supported by educational programs that use new educational technologies, such as KIBO and ScratchJr. These are: content creation, creativity, communication, collaboration, community building and choices of conduct. Each of these "6 Cs" guided the curricular activities, and choice of materials used in the summer programs sessions. These six Cs also represent how engagement was measured (see following section). For example, activities were made with goals in mind to enable content creation with the technologies, but were also opened ended to encourage creativity. At the end of each activity students participated in a Tech Circle where they could communicate and share their ideas and build a community around their projects. Children worked in an open space where they could see other's projects and could easily collaborate and communicate with them, but also tested their ability to behave properly when interacting with other children and the technology they were using.

The activities for the week followed a Safari theme, but due to the nature of each technology the activities themselves differed from one camp to another. For example, while the ScratchJr group programmed animals to run a race onscreen using speed and motion blocks in Activity 1, the KIBO group could not run race because KIBO's language does not contain speed blocks. Instead, the KIBO group programmed their robots to move like the animal of their choice. Both camps focused on teaching participants the same computational thinking concepts. Table 1 outlines the five major activities in each of the camps. The first four activities were each one hour to an hour and a half long and the final project lasted for three hours.

Table 1. KIBO and ScratchJr curriculum

Activity	KIBO	ScratchJr
1	**Animal Movement: Sequencing** Each child picks and animal and programs their robot to move like that animal. They can then decorate their robot to match the animal they select.	**Animal Race: Sequencing** Children pick 3 safari animals and program each of them to run a race at different speeds. Children have an opportunity to create animals and backgrounds.
2	**Baby Animals: Sequencing** Students will decorate and program a baby animal that moves only when it hears the parent animal's voice.	**Animal Story: Sequencing** Children will create a 1-4 page story relating to a safari. They will program several characters to perform different actions on each screen of the story.
3	**Lost Animal: Repeats** Students must use repeats to help their animals move across a series of paths to get from their current location back to their homes.	**Lost Animal: Repeats** Students must use repeats to help their animals move across a series of paths to get from their current location back to their homes.
4	**Survival Game: Conditionals** Children create programs that include sensors to help their robotic animals to run away from dangerous predators.	**Baby Animals: Conditionals** Students will use message sending to help baby animals return to their homes when they are called by their parents.
5	**Final Project: All skills** Students create a Safari Animal of their choice and program it to move through a habitat they make.	**Final Project: All skills** Children create a multi-page story about the journey that an animal is taking through the Safari.

In both the KIBO and ScratchJr group, the final project activity came at the end of the camp week and asked children to draw on all the computational concepts they had learned up to that point. In the KIBO group, children programmed their robotic safari animal to navigate through a habitat of their own creation. Meanwhile, in the ScratchJr group, children created a multi-page story about an animal's journey through the safari. Both projects involved some non-programming research and planning, such as choosing an animal, reading pictures books with facts about the animal, and learning about the safari environment.

Computational thinking

Both the KIBO and ScratchJr curriculum units focused on teaching children foundational computational thinking skills. Computational thinking skills are being defined based on aspects of Brennan and Resnick's (2012) Computational Thinking Framework that correspond with young children's developmental ability. The concepts measured in this study include: sequencing, repeats, conditionals, and debugging (Table 2). At the end of the weeklong session, students had to individually complete a Solve-Its task that tests these skills (Appendix A). The "Solve-Its" were developed to examine young children's knowledge of foundational programming concepts (Strawhacker & Bers, 2015; Strawhacker, Sullivan, & Bers, 2013).

Table 2. Definition of computational thinking skills assessed

Skill	Definition
Sequencing	A series of steps that determine the order in which actions are performed.
Loops	A mechanism for running the same sequence several times.
Conditionals	Making decisions based on certain factors or events.
Debugging	Fixing syntactical errors in a program.

During the assessment, participants completed two different types of tasks. First, children completed a series of tasks called the "Solve-Its" which involved listening to three different stories or songs being read or sang aloud by a researcher. After listening to the story or song, and the Solve-Its prompt children to arrange paper blocks into a sequential program that matched what they heard. The Solve-It tasks were developed by to target areas of foundational programming ability and basic sequencing skills (Strawhacker & Bers, 2015; Strawhacker, Sullivan, & Bers, 2014).

Figure 4. Sample child-completed wheels on the Bus Solve-It

Each Solve-It task tested one of the computational thinking skills, with the exception of debugging. For example, one KIBO Solve-It task used the song *The Wheels on the Bus* (see Figure 4). The researcher prompted the children by saying, "Do you know the song, the Wheels on the Bus? I know when we sing that song, the wheels spin around on the bus so many times! Let's sing the song together, and count how many times the wheels spin!" After singing one verse of the song with the kids, the researcher asks, "how many times did we count the wheels spinning? [pause for answers]

That's right, four times! I want to make a robot that is a bus, and I want the wheels to spin around four times, just like in the song. Can you imagine the program my robot needs? Can you make the program using the paper blocks we're passing out now"? At this point children were given several minutes to create their program on paper before moving on to the next task.

Once students completed the three programs there were given a fourth story, or Solve-It. This time instead of being given blocks to create a program, they were given a program that was incorrect. The task this time was to circle the blocks that were not in the correct location (Figure 5). Once the assessment was complete, each category was graded and given a score between 0-100 percent.

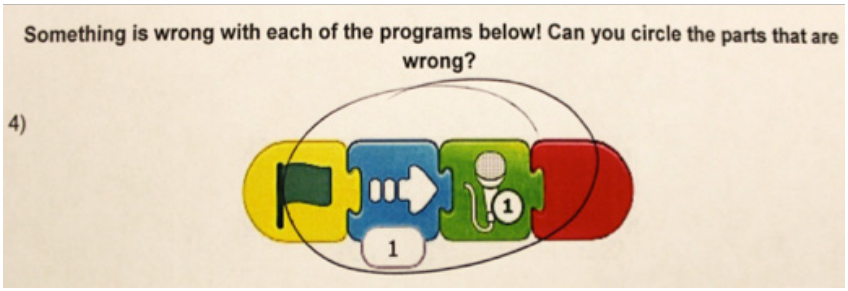

Figure 5. Sample child-completed Debugging Solve-It

Positive technological development

Learning engagement was measured using a behavioral checklist developed to complement Bers (2012) "6 C's" for Positive Technological Development. At the end of each day, counselors and a research assistant rated each child using as assessment called the "PTD Engagement Checklist" (Appendix B). The PTD Engagement Checklist prompted the counselor and researcher to look for 4-6 specific behaviors per category and mark the frequency of each behavior during the activity using a 1-5 scale (1=Never and 5=Always) (Table 3). The scores given by the counselor and the research assistant were averaged together at the end of each day. If scores given by counselors and research assistants differed by more than a point, they would discuss and come to a consensus. The scores given across each of the five days were then averaged together so each child had one score per category for the week. In addition to numerical data, they also provided written notes about general behaviors observed during.

Table 3. PTD engagement categories

Category	Definition	Sample Behavior
Communication	The process of using technology to exchange thoughts and opinions.	Student is exchanging ideas with others
Collaboration	Working with others toward a shared goal or task.	Students is giving help to others and helping them understand materials
Community-Building	Using technology to enhance the community around you.	Students is volunteering to share work with others during Circle Time
Content Creation	Making ideas come to life using technology.	Student knows how to use the technology to make a project
Creativity	Using technology in a new and unexpected way.	Student is using technology in an unexpected way
Choices of Conduct	Making conscious choices about one's behavior when using technology.	Student is following the classroom rules

RESULTS

For both the Solve-It tasks and the PTD Checklists, basic descriptive statistics were calculated as well as statistical comparisons between the two groups (KIBO and ScratchJr). On average, the children in this study were highly successful in mastering the basic computational skills taught to both groups. Detailed analysis is presented in the following sections.

SOLVE-ITS

Overall, children demonstrated a fairly high mastery of the basic computational thinking concepts assessed in the Solve-Its including: sequencing, repeat loops, and conditional statements (see Table 4). For each of these tasks, there was an overall mean score of 83 or higher (out of a possible 100). However, when it came to debugging, the mean score was much lower than the other tasks (55.56), indicating this may have been a more challenging skill for children to master.

When looking at these descriptive statistics, we can see that the KIBO group performed better on average on every Solve-It task (see Figure 6). For both groups, Debugging was the most challenging concept, although children in the KIBO group scored much higher than those in the ScratchJr group.

Table 4. Solve-Its mean scores

Solve-Its Descriptive Statistics

KIBO or ScratchJr		N	Minimum	Maximum	Mean	Std. Deviation
KIBO	Sequencing	13	50	100	96.15	13.868
	Repeats	13	25	100	90.38	21.743
	Conditional	13	50	100	90.38	16.261
	Easy Debugging	13	0	100	76.92	43.853
ScratchJr	Sequencing	14	25	100	71.43	35.161
	Repeats	14	0	100	88.10	28.063
	Conditional	14	0	100	84.29	30.562
	Easy Debugging	14	0	100	35.71	49.725

Note. One child was absent, so total sample of KIBO and ScratchJr groups is N=27

A 1-Way ANOVA was performed to examine whether interface (graphical or tangible) had a statistically significant effect on students' performance on each of the following tasks: Sequencing, Repeat Loops, Conditional Statements, and Debugging. These four tasks were selected for ANOVA analysis because each one targets discrete computational thinking concepts.

Of the four Solve-Its there was no significant impact for interface type on students' performance on the Repeat Loops or Conditional Statements tasks (p>.05). There was a significant main effect for interface on students' performance on the Sequencing task (F(1,25) = 5.605, p = .026). On this task, students in the tangible KIBO group (mean=96.15) scored significantly higher than children in the graphical ScratchJr group (mean=71.43). There was also a significant main effect for interface on children's performance on the Debugging task (F(1,25) = 5.182, p = .032). Once again, children in the tangible KIBO group (mean=76.92) scored significantly higher than children in the graphical ScratchJr group (mean=35.71). These findings indicate that children in the tangible KIBO group mastered the skills of Sequencing and Debugging significantly better than students in the ScratchJr group.

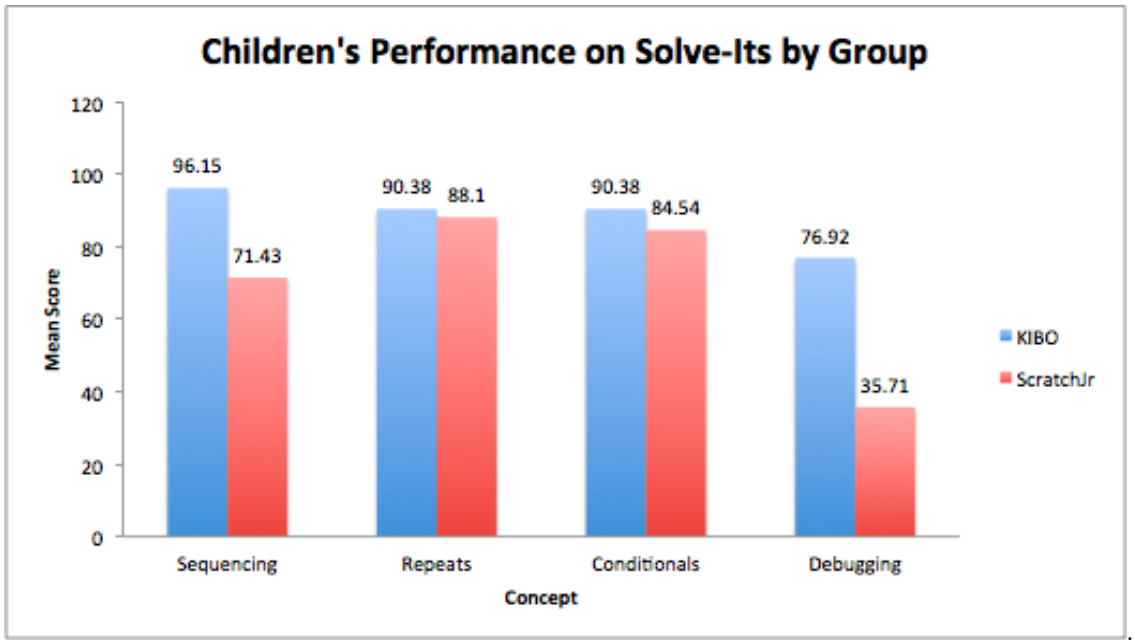

Figure 6. Children's performance on Solve-It by interface

PTD Engagement

Overall, children had fairly high PTD scores across all six Cs in both the KIBO and ScratchJr groups (Communication, Collaboration, Community Building, Content Creation, Creativity, and Choices of Conduct). This might be due to the fact that the curriculum was designed inspired by the PTD framework. The scores within each category ranged from 3.47-4.45 out of 5 (3.47-4.13 for KIBO and 3.78-4.45 for ScratchJr). An Overall PTD score was calculated based on averaging the mean scores across the six Cs. The KIBO and ScratchJr groups had very similar overall scores (3.83 for KIBO and 3.99 for ScratchJr). An Independent Samples *T*-test was calculated to confirm there was no significant difference between the two groups' overall scores. Results show that there was no significant difference ($p>.05$).

Collaboration

Children's collaborations per hour were totaled into an average collaborations score at the end of the week. Both groups demonstrated a high number of collaborations, with a mean score of 14.16 in the KIBO group and 17.14 in the ScratchJr group. An independent samples *T*-test was used to determine whether there was a statistically significant difference in collaboration score between children in the graphical ScratchJr group and children in the tangible KIBO group. Results show there was no significant difference in children's collaboration scores based on interface group ($p>.05$).

Discussion

Computational Thinking

Students in the tangible KIBO group scored higher across all four computational thinking categories in comparison to the graphical ScratchJr group. When it came to the sequencing and debugging tasks, the KIBO group performed statistically significantly better than the ScratchJr group. The difference in scores could stem from the more explicit nature of the tangible KIBO programming blocks in comparison the graphical ScratchJr interface. With KIBO children are using their hands to arrange the blocks and the motions are then translated to a robot moving in their physical space. The programming blocks and character movement in ScratchJr all occur on an iPad screen. The screen

may make it more difficult for children to discern whether or not the character they were programming followed the actions they had in mind. This result can be understood in the context of other literature that shows that young children has a developmental reliance upon physical interaction with objects to understand and explain their ideas (Bers, 2012; Froebel, 1826; Montessori & Gutek, 2004; Piaget, 1959; Vygotsky, 1978).

The sequencing score difference could also be a result of the interface through which the test was administered. The Solve-Its were administered using paper cutouts that students arranged on a piece of paper. The tangible nature of the assessment may have given students using the tangible technology (KIBO) an advantage. Since sequencing was the first skill that was assessed, the lower performance could be a result of children in the graphical group getting used to the new programming interface. For the debugging task, it is important to note that children scored the lowest in this category across both groups. This is likely a result of the curriculum itself. The other three categories were explicitly taught and aligned well with the activities that children participated in. Debugging, on the other hand, was mentioned throughout the curriculum, but never the sole focus of one of the activities.

When it came to the more advanced concepts, repeats and conditionals there were no significant differences between the KIBO and ScratchJr groups on these tasks. Both of these concepts are more advanced programming concepts than sequencing, and inherently require sequencing ability to execute (Sullivan & Bers, 2015). Children in both the KIBO and ScratchJr groups scored very highly in these Solve-It categories.

The goal in working with these technologies is not only to teach the children these four specific programming skills, but to allow them to translate what they have learning into the real world and future technologies they will encounter. The abilities to complete these tasks rely math ability, problem solving, and working memory (Sullivan & Bers, 2016). These findings demonstrate the significance implementing developmentally appropriate technologies to foster growth and development of skills both within and outside of programming.

Positive Technological Development

Children using both KIBO and ScratchJr performed well on the PTD Engagement Checklist and no significant differences were shown between their scores. This shows that, given a curriculum inspired and designed by the PTD framework, both technologies used in this study are capable of fostering a learning experience that promotes positive behaviors in a developmentally appropriate way. It also suggests that both tangible and graphical technologies have the capability to promote a collaborative environment, especially when taught with the Bers (2012) 6 Cs framework in mind.

While children performed well across the categories, there were noticeable differences in the types of behaviors they experience with teach form of technology. In terms of content creation and creativity, children in the tangible group focused on the goal at hand first and once, that was complete, they moved onto exploring other ideas. They also explored the different functions within KIBO's programming language as their main creative outlet instead of focusing on the art materials around them. Those in the graphical group often got distracted by the multitude of options within the application, but eventually completed the challenge given to them. They spent time using the paint editor features that ScratchJr offers to edit and create characters and backgrounds as their way of making their projects stand out from the rest.

There was also a difference in the way that children communicated and collaborated with each other between the two groups. In the tangible group, students were able to easily look around the room and see other student's robots. This allowed them to explore what everyone was doing and to prompt them to ask one another questions and receive peer help and input on their own work. It also allowed counselors and research assistants to easily see who was on or off task, and to see who needed help. In contrast, in the graphical group, it was much more difficult for counselors to see what children

were working on and whether or not they were off task. It was difficult to see what was on each child's iPad screen at any given time, meaning that both children and counselors needed to go out of their way to find out what everyone was working on and to ask questions. Finally, it put more responsibility on the child to ask for help, since the adults could not always tell if there were any problems with the kids' programs.

The final difference was in the general ambiance of the room. In the tangible group, children were often moving around with their robots or going over to other groups to explore their projects. The children were also generally on the louder side, especially when sharing their projects in the tech circle. They seemed very engaged and eager to share their learning. With the graphical group, children were generally very quiet and respectful in the traditional classroom sense. They were often either hyper-focused on their own work, or on the people close to them. Students only occasionally walked around to explore other people's projects. It was clear from these observations that both groups demonstrated positive conduct and community building, but it different ways.

Limitations

There are limitations to conducting a study during a summer program that parents have to voluntarily register and pay for. The biggest limitation was the self-selected nature of the children and parents. Most children did not need to be convinced that learning about programming was important and they were instantly excited to jump into the activities. This makes the group of students analyzed different than the average classroom. Another limitation was the gender demographics of students who registered for the program. In the tangible KIBO sessions, there was a significantly higher number of boys ($n = 10$) than girls ($n = 2$) registered. In the graphical ScratchJr session, there was a higher number of girls ($n = 8$) than boys ($n = 6$) registered. In a study by Horn et al. (2012), it was reported that the tangible interface seemed to appeal equally to girls and boys (in contrast to the graphical one, which was more appealing to boys). This finding was contrasted by enrollment numbers in summer camps where this study took place, which had more girls in the graphical programming camp than the tangible programming camp. Research also shows that that gender may sometimes impact young children's mastery of advanced computational thinking skills (Sullivan & Bers, 2013; Sullivan & Bers, 2016). Future research should continue to examine how gender may influence children's preference of different interfaces and their mastery of computational thinking concepts with larger sample sizes.

Other limitations came from the specific technologies that were used during this study. While the tangible KIBO robot and graphical ScratchJr iPad application both teach young children computational thinking skills, they have some differences outside of their user interfaces. For example, when a child creates a syntactically incorrect program with KIBO the robot makes a noise indicating something went wrong, and the robot does not perform any actions given in the program. It provides feedback. With ScratchJr, if the characters are given any programming blocks the character will execute them. It is up to the child to realize that what they wanted to happen in the program does not match the actions of the character on the screen. Another limitation came from access to each of the technologies. During the KIBO program, pairs of students each shared one robotics kit. For ScratchJr each child has access to their own iPad where they could easily work individually.

Future Research

While this study has its limitations, it provides pilot data to examine the differences in how children interact with tangible and graphical technologies. The results show that students clearly gained skills and had positive experiences using either technology, but that their experiences using each were different. Conducting similar research in a classroom setting with more diverse groups of students will be a next steps. Furthermore, future studies will look at graphical and tangible technologies other than ScratchJr and KIBO. This would give insight on whether the results found are specific to affordances of tangible and graphical interfaces, or simply to the two technologies themselves.

CONCLUSION

New technologies for introducing computational thinking to young children are growing in prevalence. This study shows that children ages 4-7 can learn basic computational thinking skills when given developmentally appropriate tools and proper curricular instruction. The tangible KIBO Robotics kit and the graphical ScratchJr iPad application allow children to learn sequencing, repeating, conditionals, and debugging, all of which are basic computational thinking skills. They also allow children to learn in a way that engages them while encouraging positive behavior for socio-emotional development. This study also provides preliminary evidence that a technology's user interface has an impact on the experiences that children have. Tangible and graphical interfaces each have qualities that foster different types of learning. It is up to teachers and parents to decide which work better for their students and children.

REFERENCES

American Academy of Pediatrics. (2003). Prevention of pediatric overweight and obesity: Policy statement. P*ediatrics*, *112*, 424-430.

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.

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

Bers, M. U. (2010). The TangibleK Robotics Program: Applied computational thinking for young children. *Early Childhood Research and Practice, 12*(2).

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

Bers, M.U. (2017). *Coding as a playground: Programming and computational thinking in the early childhood classroom*. London, Routledge Press.

Bers, M., Ponte, I., Juelich, K., Viera, A., & Schenker, J. (2002). Teachers as designers: Integrating robotics in early childhood education. *Information technology in childhood education*, *1*, 123-145.

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada*. Retrieved from http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Brewster, C., & Fager, J. (2000). *Increasing student engagement and motivation: From time on task to homework*. Portland, OR: Northwest Regional Educational Laboratory.

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.

Cheng, L. K., Der, C. S., Sidhu, M. S., & Omar, R. (2011). GUI versus TUI: Engagement for children with no prior computing experience. *Electronic Journal of Computer Science and Information Technology (eJCSIT), 3*(1), 31–39.

Common Sense Media. (2013). *Zero to eight: Children's media use in American 2013*. San Francisco: Common Sense Media.

Elkin, M., Sullivan, A., & Bers, M.U. (2016). Programming with the KIBO Robotics Kit in preschool classrooms. *Computers in the Schools, 33*(3), 169-186.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6 year old kindergarten children in a computer programming environment: A case study. *Computers & Education*, *63*, 87-97. DOI:10.1016/j.compedu.2012.11.016

Finn, J. D., & Rock, D. A. (1997). Academic success among students at risk for school failure. *Journal of Applied Psychology 82*(2): 221–34.

Froebel, F. (1826). *On the education of man (Die Menschenerziehung*), Keilhau/Leipzig: Wienbrach.

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

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

International Society for Technology in Education and The Computer Science Teachers Association. (2011). *Operational definition of computational thinking for K-12 thinking*. International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA). Retrieved from http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf

Jablon, J. R., & Wilkinson, M. (2006). Using engagement strategies to facilitate children's learning and success. *Beyond the Journal, Young Children on the Web*. https://www.naeyc.org/files/yc/file/200603/JablonBTJ.pdf

K12 Computer Science Framework (2016). Retrieved from http://www.k12cs.org

Kazakoff, E. R., & Bers, M. U. (2011, April). *The impact of computer programming on sequencing ability in early childhood.* Paper presented at the American Educational Research Association Conference (AERA), Louisiana, New Orleans.

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. U. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal, 41*(4), 245

Lee, K., Sullivan, A., & Bers, M. U. (2013). Collaboration by design: Using robotics to foster social interaction in Kindergarten. *Computers in the Schools*, 30(3), 271-281.

Manches, A., & Price, S. (2011). Designing learning representations around physical manipulation: Hands and objects. *Proceedings of 10th International Conference on Interaction Design and Children* (pp. 81-89). Ann Arbor, MI: ACM Press. DOI:10.1145/1999030.1999040

Marks, H. M. (2000). Student engagement in instructional activity: Patterns in the elementary, middle and high school years. *American Educational Research Journal 37*(1): 153-84.

Marshall, P. (2007). Do tangible interfaces enhance learning? *First International Conference on Tangible and Embodied Interaction* (pp. 163-170). New York: ACM.

Montessori, M., & Gutek, G. L. (2004). *The Montessori method: The origins of an educational innovation: including an abridged and annotated edition of Maria Montessori's the Montessori method.* Lanham, MD: Rowman & Littlefield Publishers.

Perlman, R. (1976). Using computer technology to provide a creative learning environment for preschool children. *AI memo 360* (No. LOGO-24; pp. 1–31). Cambridge, MA: MIT.

Piaget, J. (1959). *The language and thought of the child* (3d ed.). New York: Humanities Press.

Portelance, D.J., & Bers, M. U. (2015). Code and tell: Assessing young children's learning of computational thinking using peer video interviews with ScratchJr. *Proceedings of the 14th International Conference on Interaction Design and Children* (IDC '15). Boston, MA: ACM.

Portelance, D. J., Strawhacker, A., & Bers, M. U. (2015). Constructing the ScratchJr programming language in the early childhood classroom. *International Journal of Technology and Design Education*, 1-16. DOI:10.1007/s10798-015-9325-0

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60-67. DOI:10.1145/1592761.1592779

Resnick, M., Martin, F., Berg, R., Borovoy, R., Colella, V., Kramer, K., & Silverman, B. (1998). Digital manipulatives. *Proceedings of the Computer Human Interaction Conference (CHI98), Los Angeles.*

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

Strawhacker, A., Lee, M., Caine, C., & Bers, M. U. (2015). ScratchJr Demo: A coding language for kindergarten. *Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15).* Boston, MA: ACM.

Strawhacker, A., Sullivan, A., & Bers, M. U. (2013). TUI, GUI, HUI: Is a bimodal interface truly worth the sum of its parts?. *Proceedings of the 12th International Conference on Interaction Design and Children (IDC'13)* (pp. 309-312). New York, NY: ACM.

Sullivan, A., & Bers, M. U. (2013). Gender differences in kindergarteners' robotics and programming achievement. *International Journal of Technology and Design Education*, 23 (3), 691-702.

Sullivan, A., & Bers, M. U. (2015). Robotics in the early childhood classroom: Learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education*. DOI:10.1007/s10798-015-9304-5

Sullivan, A., & Bers, M. U. (2016). Girls, boys, and bots: Gender differences in young children's performance on robotics and programming tasks. *Journal of Information Technology Education: Innovations in Practice, 15,* 145-165.

Sullivan, A., Elkin, M., & Bers, M. U. (2015). KIBO Robot Demo: Engaging young children in programming and engineering. *Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15).* Boston, MA: ACM.

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

Vygotsky, L. (1978). Interaction between learning and development. In M. Gauvain & M. Cole (Eds.), *Mind and society* (pp. 79–91). Cambridge, MA: Harvard University Press.

Wilson, C., Sudol, L. A., Stephenson, C., & Stehlik, M. (2010). *Running on empty: The failure to teach K-12 computer science in the digital age.* New York, NY: The Association for Computing Machinery and the Computer Science Teachers Association.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

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.

APPENDIX A

SOLVE-IT ASSESSMENTS

For each assessment, you will read the story once before handing out blocks for kids to make a program. After blocks are given out read the story one more time. Wait 15-30 seconds then read the story again. Have kids raise their hands when they are done and you can tape down their blocks.

Solve-Its Category	KIBO Script	ScratchJr Script
Sequencing	**Puddle** *In an animated voice*: "This story is called Puddle. Do you know what a puddle is? Sometimes a puddle is made of water, or mud. Do robots like water? [*Wait a moment*] No, usually water can break a robot, which is not good at all! I want to make a program that lets my robot dry itself off after it accidentally moves into a puddle. First, my robot will turn on, and then it will move straight ahead – but OOPS! My robot is in a puddle! It's going to make a noise – Beep! – as if it is saying 'Oh no!' Then, I want the robot to shake itself dry – shake! – and finally, turn off!" "Can you imagine the program my robot needs? Are you ready to try to make the program for my robot?" * **Solution: Begin, Forward, Beep, Shake, End**	**Puddle** *In an animated voice*: "This story is about a Cat who walking into a puddle. Do you know what a puddle is? Sometimes a puddle is made of water, or mud. Do cats like getting wet? [*Wait a moment*] No, they like to drink water but they don't like getting wet. I want to make a program that lets my cat jump out after it accidentally moves into a puddle. When I touch the cat, it will move straight ahead – but OOPS! The cat is in a puddle! It's going to make a noise – Meow! – as if it is saying 'Oh no!' Then, I want the cat to finally, hop out of the puddle onto dry ground. *Hand out blocks after story is read. "Can you imagine the program my cat needs? Are you ready to try to make the program for my cat?" **Solution: Begin, Forward, Sound, Hop, End**
Repeat	**Wheels on the Bus** *In an animated voice*: "Do you know the song, the Wheels on the Bus? [*Wait a moment*] I know when we sing that song, the wheels spin around on the bus so many times! Let's sing the song, and count how many times the wheels spin!" [*With children, sing one verse of song, while holding up one finger to count each time "round and round" is sung*] The wheels on the bus spin round and round four times! I want to make a robot that is a bus, and I want my wheels to spin around four times, just like in the song. How would I do that?" "Can you imagine the program my robot needs? Are you ready to try to make the program for my robot?"	**Wheels on the Bus** *In an animated voice*: "Do you know the song, the Wheels on the Bus? [*Wait a moment*] I know when we sing that song, the bus driver tells people to move on back so many times! Let's sing that part of the song, and count how many times the driver says "Move on Back"!" [*With children, sing one verse of song, while holding up one finger to count each time "Move on back" is sung*] The driver on the bus says "Move on Back" four times! I want my driver to say "Move on Back" four times, just like the song. How would I do that." "Can you imagine the program my driver needs? Are you ready to try to make the program for the driver?"

Conditional	**Surprise Happy Birthday** "Now our robot wants to surprise its sister for her birthday! The robot is only going to sing "happy birthday" if its sister is in the room. So our robot is waiting, and if the sister is nearby, the robot will sing."	**Surprise Happy Birthday** "My friend Bob wants to surprise Emma for her Birthday. Emma sends Bob a message that she is about to arrive at his house. When bob receives the message he appears from behind a table and sings Happy Birthday to her."

For the circle the block stories, the same rules follow. Read the story once, and then tell kids that they can circle the block (or blocks) they feel are in the incorrect spot. Read the story again, wait 15-30 seconds then read the story one last time. Have kids raise their hands when they are done.

Solve-Its Category	KIBO Script	ScratchJr Script
Easy Debugging	**Car Horn** *In an animated voice*: "This game is about a robot that is a car. Have you ever heard a car honk its horn? Can you make the 'BEEP BEEP' sound? [*Wait a moment*] I want my car robot to turn on – start the engine, vroom! Next, I want to honk the horn – Beep Beep! – to warn people that I'm about to move. Then I want my car to drive straight ahead, and then stop! And turn off." *Repeat explanation once more*. "Can you imagine the program my car needs? Are you ready to try to make the program for my robot?"	**Car Horn** *In an animated voice*: "This game is about a car. Have you ever heard a car honk its horn? Can you make the 'BEEP BEEP' sound? [*Wait a moment*] I want my car turn on – start the engine, vroom! Next, I want to honk the horn – Beep Beep! – to warn people that I'm about to move. Then I want my car to drive straight ahead, and then stop! And turn off."
Hard Debugging	*Washing Machine* In this story, my robot is actually a washing machine! Have you ever seen a washing machine shake the clothes to make them clean? First, I want my washing machine robot to turn on. Then I want it to shake and wash the clothes, and keep doing it for three minutes. Then, I want the robot to stop shaking when the clothes are clean, and to make a noise - Beep! - to let me know that it is done! Last, I want the washing machine to turn off." So, my robot will turn on, repeat shaking four times, stop shaking, beep once, and then stop. Can you make a program that matches this story? Remember, some of the block pictures can go on top of other block pictures if you want for this program."	*Merry-Go-Round* This story is about a Merry-Go-Round! Have you ever seen a Merry-Go-Round? First, I want my Merry-Go-Round to turn on. Turn the kids on ride around and around three times. Then, I want the ride to stop turning and to make a noise - Pop! - to let all the kids know that the ride is over! Last, I want Merry-Go-Round to turn off." So, the ride will turn on, repeat turning three times, stop turning, beep once, and then stop. Can you make a program that matches this story? Remember that you need to write down how many times you need to repeat."

APPENDIX B

POSITIVE TECHNOLOGICAL DEVELOPMENT (PTD) CHECKLIST

On a scale from 1 to 5 (1 = Never, 2 = Almost never, 3 = Sometimes, 4 = Often, 5 =Always, N/A = Not Applicable), how often do students do the following?
Communication
Students are exchanging ideas with others
Students feel comfortable seeking help and asking questions
Students ask and respond to questions relevant to the learning happening
Students are eager to share ideas with others
There is time allocated in the schedule for children to talk with each other
The arrangement of classroom allows for children to talk with one another (ex. desks are arranged so that students are facing one another)
Collaboration
Students are giving help to others and helping them understand materials
Students are receiving help from others and appreciating it
Students are borrowing or lending materials
Students are working together towards a common goal
There is time in the schedule for students to work together
There are a variety of spaces in the classroom where two or more students can work together
Community Building
Time is allotted in the schedule for students to share their projects with peers (Circle Time)
Time is allotted in the schedule for students to share their projects with families, school administrators, etc. (Open House)
Students are volunteering to share work with others during Circle Time
Students are volunteering to share their work with families, school administrators, etc. during Open House
Students are creating projects to solve a social problem (ex. Project to help the environment, save an animal, teach younger kids, etc.)
Student are participating in community-related tasks (ex. helping with clean-up, set up, etc.)
Content Creation
Students know how to use the technology to make a project
Students can create a functional program for their robot/character
Students are interested and enthusiastic about their project
Student are persisting in spite of obstacle or setbacks
Students know how to debug their programs

There is space in the classroom for students to test out their programs
There is time allocated in the schedule for students to learn about, practice and fix their projects
Creativity
Student are using a variety of materials (arts, crafts, etc.) or functions (ex. adding a background, editing/making a character) for their project
Student are using technology in an unexpected way
Students' projects shows unique characteristics, i.e. it is different from everyone else's
Students exhibit confidence and can initiate and complete a task with limited coaching
There are a variety of materials available for students to choose from
There is allotted time in the curriculum for students to brainstorm ideas for their projects
The projects are introduced to students as open-ended; there is more than one way to create a project
Students are given basic guidelines for their project, but there is also opportunity for them to expand beyond them
Children are having fun as they work on their projects
Choice of Conduct
Students are focused on the activity and choose to engage with it
Student are following classroom rules
Students are following rules about using technology, and they know how to use it responsively
Student are using materials responsibly
Student are showing respectful behaviors to peers and teachers
There is time in the schedule to discuss rules about using technology and how to behave in the classroom

BIOGRAPHIES

Alex Pugnali was a researcher at the DevTech Research Group in the Eliot-Pearson Department of Child Study & Human Development at Tufts University. He received a Bachelors in Human Factors Engineering from Tufts University. Alex now works at the Center for Engineering Education and Outreach also at Tufts. Alex is particularly interested in providing elementary school students from all backgrounds with access to STEAM Education in fun and creative ways. He is also interested in exploring the relationship between how technology, materials and curricula are designed and the implications they have on how students and teachers interact with them.

Amanda Sullivan PhD is a researcher at the DevTech Research Group in the Eliot-Pearson Department of Child Study & Human Development at Tufts University. Amanda is also the Associate Director of the Early Childhood Technology Graduate Certificate Program at Tufts University. She received her PhD and MA in Child Development from Tufts University where her dissertation explored young children's development of gender stereotypes about technology and engineering. Amanda's research interests include gender stereotypes, girls and STEM, curriculum development, robotics, programming, STEAM, and the arts. More on Amanda: http://ase.tufts.edu/devtech/amanda_sullivan.html

Marina Umaschi Bers PhD is a Professor at the Eliot-Pearson Department of Child Study and Human Development and the Computer Science Department at Tufts University. She heads the interdisciplinary Dev-Tech Research group which focuses on designing studying innovative learning technologies to promote positive youth development. Dr Bers received prestigious awards and has written three books: *Coding as a Playground: Programming and Computational Thinking in the Early Childhood Classroom* (2017; Routledge Press), *Blocks to Robots: Learning with Technology in the Early Childhood Classroom* (2008; Teacher's College Press) and *Designing Digital Experiences for Positive Youth Development: From Playpen to Playground* (2012; Oxford University Press). Dr Bers has an MEd from Boston University and an MS and PhD from the MIT Media Lab. More on Dr Bers: emerald.tufts.edu/~mbers01/