CrossMark

"I want my robot to look for food": Comparing Kindergartner's programming comprehension using tangible, graphic, and hybrid user interfaces

Amanda Strawhacker · Marina U. Bers

Abstract In recent years, educational robotics has become an increasingly popular research area. However, limited studies have focused on differentiated learning outcomes based on type of programming interface. This study aims to explore how successfully young children master foundational programming concepts based on the robotics user interface (tangible, graphical, hybrid) taught in their curriculum. Thirty-five Kindergarten students participated in a 9-week robotics curriculum using the LEGO WeDo robotics construction kit and the Creative Hybrid Environment for Robotic Programming (CHERP) programming language. A mixed methods data collection approach was employed, including qualitative observational data from the classrooms, as well as quantitative mid- and post-test assessments of students' programming knowledge using CHERP. The findings show little association between user interface and programming comprehension, although there may be an order-affect when introducing user interfaces. Implications for best practices when introducing programming in early childhood settings are discussed.

Keywords User interfaces for education · Tangible programming · Kindergarten · Robotics · Early childhood education

Introduction

In recent years novel interfaces for learning have been gaining widespread popularity. Specifically, there is a growing interest in tangible interfaces, defined as concrete, physical manipulatives that can directly impact a digital environment (Africano et al. 2004; Raffle et al. 2004; Tsong et al. 2012; Suzuki and Kato 1995; Zuckerman and Resnick 2003). Tangible interfaces are of special interest in early childhood education as they resonate with traditional learning manipulates, such as Montessori blocks and Froebel's gifts

A. Strawhacker (✉) · M. U. Bers
The DevTech Research Group, Tufts University, 105 College Ave, Medford, MA 02155, USA
e-mail: Amanda.Strawhacker@tufts.edu

specifically designed to learn about mathematical concepts (Bers 2008). Unlike traditional graphical user interfaces (GUI), a tangible user interface (TUI) allows the user to input digital information by manipulating a physical object rather than using a screen, keyboard, or mouse. Furthermore, Piagetian theory of the early 20th century provides a theoretical basis for promoting their use by proposing that interaction with the physical world is a primary learning avenue for young children (Piaget 1959).

Digital interfaces become powerful learning tools when children become creators instead of just consumers of digital content (Resnick et al. 1998). When programming, children explore the foundational concepts of sequencing, pattern recognition, and cause-and-effect relationships that are already represented in Kindergarten curricula when children learn how to tell a story from beginning to end, sequence numbers and letters, and order objects by size, shape, or color (Kazakoff et al. 2013). Tangible programming interfaces are especially well-suited for introducing young children to programming as they can match their developmental skill-set. They leverage children's natural kinesthetic learning strategies to make abstract concepts accessible and intuitive (Xu 2005). Researchers have found that it is possible to design programming languages that children can easily engage in by "simplifying the grammar and representing mechanisms more concretely" (Wyeth 2008). This has resulted in several new languages developed specifically for young children: Cricket (Resnick et al. 1998), AlgoBlock (Suzuki and Kato 1995), *curlybot* (Frei et al. 2000), Topobo (Raffle et al. 2004), and Electronic Blocks (Wyeth 2008), to name a few of the most widely-researched. Despite the hype generated by new tangible interfaces, more research is needed to understand their potential for promoting learning, and what children gain from tangible programming languages.

This paper presents results of a pilot study conducted in three Kindergarten classrooms with three different kinds of interfaces for teaching children how to program a robot: a *graphical user interface (GUI)*, a *tangible user interface (TUI)*, and an interface in which users can switch freely between tangible and graphical input, which in this paper are referred to as *hybrid user interface (HUI)*. These interface styles will be described in more depth later in the paper, but essentially, a graphical interface presents visual images and icons instead of lines of text to a user or programmer. A "graphical interface" typically refers to a computer screen running a pictoral operating system like Windows or Mac. Tangible interfaces are built to allow users to engage with digital devices in the same way, except that instead of 2-D visuals and icons, the control mechanism is a three-dimensional system of some kind. These may take the form of digital blocks, reactive furniture, organic materials with built-in sensors, etc. Because tangible interfaces are relatively new, it is difficult to identify well-known example. The key difference between tangible programming interfaces and regular digital tangibles (i.e. an electric on/off switch) is the power of the interface to create sequences of commands. Hybrid interfaces are even more difficult to describe, especially since they are defined differently in various research fields. For the purposes of this paper, hybrid interfaces are any interface environment in which users can choose and switch between graphical computer screens and tangible wooden blocks to complete their programming sequences. All of the interfaces described here are digital interfaces, since they are used to program electronic, motorized Lego constructions.

Research on developmental learning theory has shown that different tools and experiences may make different concepts more salient for a learner (Manches and Price 2011). Using hands and objects instead of mental models may actually change the way a child remembers and retrieves the information taught. Does the different interface impact children's comprehension of programming concepts? Are certain interfaces more conducive to learning certain kinds of concepts? Does the order in which interfaces are

introduced affect children's programming learning? Is programming comprehension improved by learning with multiple interfaces, or just one? These are some of the questions explored in the research presented in this paper.

Young children's learning: tangible programming in context

Children between the ages of 4–6 years old are learning how to understand, represent, and remember objects in their mind without having the object in front of them (Piaget 1959). They still struggle with abstract concepts like perspective-taking, and mental modeling, so they often need to rely on physical representations and objects to help them formulate, test, and revise their ideas about how their world works (Ackermann 1996; Piaget 1952). It seems like a daunting list of cognitive concepts to master before first grade. However, Kindergarten children have a special learning tool to aid in this development: play. In particular, play that involves the manipulation of physical objects that symbolize other objects or ideas lets children explore the rules and systems of larger abstract concepts in a variety of educational domains (Piaget 1952; Bers 2008).

While playing, children engage with the physical world using their five senses (Vygotsky 1978). Friedrich Froebel, the inventor of the modern Kindergarten, understood this when he created his "gifts," toys developed specifically for little hands to construct and dismantle (Froebel 1826). These small tools allow children to model structures they see in their daily lives, or to build entirely new ones of their own design. Maria Montessori later expanded on Froebel's concept of physical learning tools, and developed her own classroom experience that focused on using tangible manipulatives in modeling, thus making abstract ideas more salient (Montessori and Gutek 2004). This line of work was extended in recent years by researchers working with digital manipulates (Bers 2008). For example, Zuckerman et al. (2005) explain that in addition to providing the sensory engagement that young students need, tangible manipulatives also provide easy entry-points for novice learners or students with learning disabilities, as well as a natural platform for group work and shared control, which fosters discussion and social negotiation—important aspects of learning-related play (Vygotsky 1978). Perhaps this helps understand why it is so common to see tangible tools and manipulatives in Kindergarten classrooms in the form of puzzles, counting cubes, and colorful plastic shapes, among many others.

As mentioned above, manipulatives have shown over time to be uniquely supportive of young children's learning and creative expression. Thus, it seems that the logical next step is to expand the physical interactions to the digital world. However is there an impact of tangible interfaces on young children's learning?

Although there is still relatively little research about young children and tangible interfaces, there is evidence that older learners can gain different benefits from tangible interfaces than from graphical ones. For example, a study comparing computer and hardwood interfaces of a program designed to teach middle school students about the human body found that the graphical interface was more efficient and promoted faster learning of new concepts, while the tangible interface promoted deeper engagement and prolonged focus on exploring new anatomical structures (Ploderer 2005). In an experiment comparing physical and screen-based puzzles, children aged 7–9 years completed the tangible puzzles faster and with more direct peer collaboration than the computerized ones (Xie et al. 2008). Studies on adult learning in medical technician settings have also shown that tangible interfaces may be more intuitive to master, more supportive of problem solving behaviors, and better at fostering spatial awareness, especially for users who have low spatial

cognition (Quarles et al. 2008; Kim and Maher 2006). These findings are supported by embodied cognition and gesture research, which show that gestures aid thinking by recalling prior manipulation of materials (Manches and Price 2011). However, Manches and Price caution that it is still unclear whether encouraging manipulation of materials necessarily results in learning gains. They suggest that when investigating the benefits of tangible interfaces, specific learning domains and objectives must be considered (Manches and Price 2011).

While there is a growing focus on investigating tangible interface learning, there is still little empirical evidence that tangible interfaces offer educational benefits compared to graphical or other digital methods (Horn et al. 2012; Marshall 2007). In a study by Horn, et al. (2012) on a museum exhibit using Tern, a tangible/graphical robotic programming language, tangible interfaces were shown to be more appealing to children (defined as participants aged "16 and under") and more useful for fostering collaboration. These findings have been echoed by similar studies (Cheng et al. 2011; Manches and Price 2011). However, Horn et al. (2012) reported unique findings that the tangible interface seemed to appeal equally to girls and boys (in contrast to the graphical one, which was more appealing to boys), and that users had similar levels of understanding and program complexity regardless of whether they were using the TUI or the GUI version of the software.

These findings inform the work presented in this paper. Furthermore, this paper sets to do a control study comparing three different types of interfaces used by young children, at an age in which research has already proved the impact of traditional learning manipulatives. Do we see similar results when children use tangible programming interfaces to control the behavior of a robot? What happened when children are able to choose the interfaces they want to work with? Would different interfaces influence children's understanding of the content they learned, or of the problem solving techniques inherent in the learning process? This study is designed to explore these issues. The next section of the paper describes the specific research questions.

Blocks versus computers: What can children gain?

The present study is inspired by the work of Horn et al. (2012) that first explored initial interactions of mixed-age participants (under 16 years) with a tangible and graphical programming language in an informal learning setting: a museum. This study explores similar questions, by focusing on the impact of the interface on comprehension about programming, and expands by looking at changes in understanding over time in the context of a formal learning setting: a Kindergarten classroom. Both tangible and graphical interfaces were introduced to Kindergarteners in different combinations throughout a 9-week robotics programming curriculum. Kindergarten classrooms were chosen specifically because of this age group's developmental reliance upon physical interaction with objects to understand and explain their ideas (Piaget 1959; Froebel 1826; Montessori and Gutek 2004; Vygotsky 1978; Bers 2012)

Two related but distinct overall research questions drove this work: how well children learn *with* technology (i.e. programming languages in the context of robotics), and how well children learn *about* technology (i.e. programming languages in the context of robotics). For example, when we examine a child interacting with a programming interface, we are equally curious about what she does with it and how she engages with it, and also how complex or functional her programmatic creation is, and how closely it aligns with her stated goal. To that end, our formal research questions are:

1. Does the interface style (i.e. tangible vs. graphical vs. hybrid) of the programming language affect a child's comprehension of programming concepts in the context of a structured Kindergarten classroom?
2. Given the option to switch between interfaces, are there any patterns in children's interface choices and the impact on comprehension of programming concepts?

Study design and method

This study exposed three Kindergarten classrooms to different programming interface styles (tangible vs. graphical vs. hybrid) in order to observe differences in children's learning about programming. This section focuses on the different elements in the study design. First, it presents the curriculum about computer programming, an adapted version of the robotics curriculum developed by the DevTech Research Group (Sullivan et al. 2013). The same curriculum was used to teach all participant students regardless of the interface condition. Second, it describes the three different interface conditions experienced by each classroom. Third, it provides descriptions of the participant demographics. Finally, assessment instruments are presented.

The robotics curriculum

During the study, each of the three Kindergarten classrooms participated in a total of 13 h of robotics lessons. The lead researcher taught the 9-week robotics curriculum with the help of research assistants, using technology lesson guides developed at the DevTech Research Group (Sullivan et al. 2013). This curriculum is designed following the positive technological development framework (Bers 2012). The curriculum integrated the learning of robotic programming with the school's standard science curriculum on animals and habitats. Children were tasked with building a robotic animal and programming it to act the way the animal might act in real life, i.e. sleeping, playing, looking for food, etc. Classes followed the school's normal Kindergarten schedule, with small-group work (2–3 children), broken up by short large-group activities (Kostelnik and Grady 2009).

The three core programming concepts in the curriculum are: sequencing, repeat loops, and creative programming, or programming with a meaningful goal in mind. These concepts were introduced sequentially, as each one builds upon the last. The concept of sequencing introduced the idea that order matters when giving instructions or actions to a robot. Most specifically, the order of programming commands will affect the order of actions the robot executes. The repeat loops lessons explored actions and sequences that repeated within a program, essentially creating small programs inside of longer ones. Finally, the unit on creative programming encouraged children to choose an imaginative goal for their robotic animal (i.e. find food or travel to its habitat) and build a program to execute their goal (see Fig. 1). This allowed children to explore all programming concepts learned earlier, and to apply them toward a personally meaningful project.

Every unit taught contained a "buffer lesson," during which children were allowed to explore with the programming and robotic materials with no directed lesson goal (see Table 1). This was designed to give children time to absorb concepts, and maintain focus during more structured activities. After the repeat loops unit and the creative programming unit, assessments were administered (the reasoning for this timing is explained in later
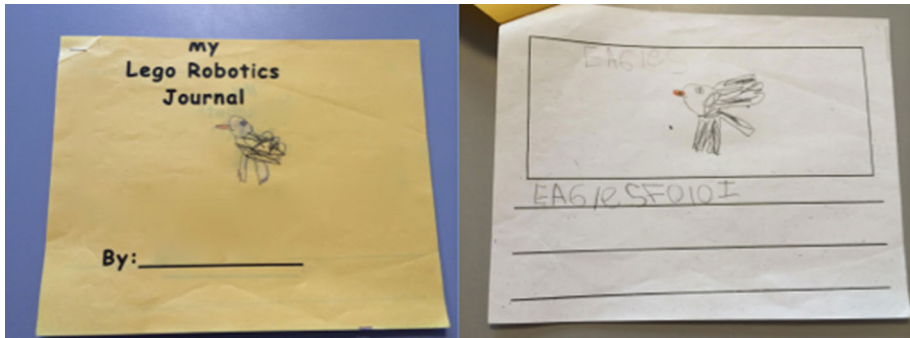
Fig. 1 Sample child's engineering journal. Children used journals to track their animal research and engineering designs. This student chose to study eagles, and practiced writing "eagles fly" in her notes

sections). Assessments were designed to target children's understanding of each programming concept.

As mentioned before, this robotics programming curriculum was integrated with the core science Kindergarten curriculum that focuses on animal habitats and characteristics. Children were asked to use their robotic materials to build and program a robotic animal in its habitat. This happened during the sessions that focused on the open-ended creative goal.

Robotic tools and programming interfaces

All children used Lego® Education WeDo Construction Sets to build their robots and the Creative Hybrid Environment for Robotic Programming (CHERP) programming language to program the robot's behaviors (Bers 2010). CHERP is composed of color-coded, high-level programming instructions, such as "forward," "spin," and "beep." CHERP code may be written using an on-screen computer interface or tangible wooden blocks. The GUI allows children to click-and-drag picture icons of instructions into a list, or sequence, for the robot to act out. The TUI allows children to put together interlocking wooden blocks, with stickers of the CHERP picture icons on the sides (see Fig. 2). CHERP is a direct descendant of the Tern language used by Horn, et al., in the museum investigation described early on (Bers and Horn 2010; Horn et al. 2012).

Participating classrooms

Participants were Kindergarten students in a public early education school in the city of Boston. The school hosts pre-kindergarten through third grade students from mainly urban areas, with populations characterized by low socio-economic status. At the time that the intervention began, the school had no formal robotics or engineering component to its curriculum. Massachusetts State demographic and census information reports that the school is 72 % Hispanic, 69 % Limited English Proficiency, 65 % Free or Reduced Lunch, 15 % Special Education (MA DOE 2011).

Three classrooms participated, and these were purposefully selected to meet three requirements. First, the children all had to be the same age (5–6 years) so that researchers could draw comparisons across the sample, controlling for approximate developmental skill level. Second, children with previous formal robotics education were not accepted

Table 1 Curriculum content, by lesson day

Lesson	1	2	3	4	5
Unit 1 content	Intro	Programming: sequencing			Buffer day: explore
Lesson		6	7	8	9
Unit 2 content		Programming: repeat loops			Assessment day
Lesson		10	11	12	13
Unit 3 content		Programming: creative goal			Assessment day

Fig. 2 CHERP interfaces and LEGO WeDo robotic kit. Computer screen shows graphical CHERP interface, and blocks represent tangible interface

into the study (this exempted one Kindergarten classroom from the same school), since their prior experience could alter their reactions and engagements. Finally, the children receiving the intervention were required to participate as a cohesive classroom unit. This was intended to provide children with a safe social environment for their learning, an idea borrowed Vygotsky's (1978) theories on social context in the development process.

There were also important differences among the participating classrooms. Despite the fact that all three classrooms were part of the same public elementary school, two classrooms were taught using traditional public school methods and the third was taught Montessori-style. Since Montessori classrooms are typically mixed-age, many of those children were ineligible to be research participants. However, we needed three participant groups to address the research questions, so we elected to include in the study only those children who were age 5–6 from the Montessori class. This means that the sample size of one group (the GUI classroom, see section on Experimental Conditions) is much smaller than the other two groups.

The full-time classroom teachers were present and involved during each lesson. Teachers participated by assisting children during free-work time and managing transitions and behavioral issues during large group discussions. Only one teacher (TUI classroom) attempted to co-lead lessons later in the curriculum. The HUI classroom teacher expanded the robotics class content (animals and hibernation) into other domain-specific activities. For example, she led a writing activity in her classroom about planning how the children's

robotic animals would look and act. Each classroom was initially exposed to different experimental conditions, which are described in the next section.

Blocks, computers, or both?

While each participating Kindergarten classroom was exposed to the same curriculum and used the same CHERP programming language, they each used a different CHERP interface: tangible, graphical, and hybrid. These conditions were in place for the first six of the nine programming lessons in the curriculum (see earlier timeline). During the last three lessons children were introduced to both tangible and graphical interfaces, meaning all classrooms used a hybrid interface.

The conditions were chosen to let researchers follow the children's development of curriculum comprehension using quantitative assessments tools (Strawhacker et al. 2013) designed specifically to gauge children's programming knowledge. In order to provide a context for our quantitative data, observational data was collected and coded about each child's experience in the classroom.

Tangible (TUI) condition

In the tangible condition classroom (N = 14), children were given formal instruction using only the CHERP tangible wooden blocks. Because CHERP requires a computer to upload and compile the tangible program to the WeDo robot, there were two "testing stations" set up around the room, where research assistants used the computer's embedded webcam to upload and compile programs for children. Children were able to observe the research assistants while they uploaded the programs, watch their robots executing the programs they had written, and edit their programs using the wooden blocks. They were not permitted to edit the programs using the computer interface, and research assistants did not explain any part of the computer screen interface to the children. Children worked in groups of three children.

Graphical (GUI) condition

Children in the graphical condition classroom (N = 7) were given formal instruction using only the computer screen interface of CHERP. Each small group of children shared one computer for programming and for testing their robots at their work areas. Computers were small eeePC laptops, and children were allowed to use a small USB mouse to drag and drop graphical icons into the programming area. No reading or writing skills are needed for using the GUI. Children worked in groups of two to three children.

Hybrid (HUI) condition

In the third classroom (N = 14), children were given formal instruction using both the tangible and graphical versions of CHERP simultaneously. Each small group of three children had a laptop and mouse and access to wooden blocks. Children were able to build, upload, and compile programs at their work areas, and were given formal instruction on the use of both interfaces. Although the field of tangible computer interfaces might use the term hybrid in a different way (Sehyun 2006), for the purpose of this paper, we refer to

HUI, or hybrid user interface, to describe an environment where children can freely switch between two different types of interfaces at any point, and use both simultaneously to work on a programming activity.

Timeline of conditions

For the first six programming lessons, classrooms were conducted using the interface conditions described above. For last three programming lessons, all classrooms were introduced to the hybrid condition (HUI) to account for order effect (see Table 2).

There were two reasons for introducing the hybrid environment to the tangible and graphical condition classrooms. On the one hand, we wanted to explore children's interface preferences, once they were exposed to both TUI and GUI. Since CHERP was designed to be used as both a tangible and a graphical programming language, we wanted to expose children to the kind of experience they would have in a classroom in which both interfaces could be used interchangeably. On the second hand, by providing children the opportunity to switch freely between interfaces, we may also have given them a problem solving tool. Manches and Price (argue that when children face challenges in the design process, it is important that they are able to choose which tools they want to use to solve the problem, as this helps them to better understand the problem itself (Manches and Price 2011). CHERP was designed with this principle in mind.

Data collection: learning from the students

Data was collected using a mixed-methods approach, incorporating aspects of qualitative and quantitative analysis methods (Creswell 2014). Quantitative data was collected in the form of Solve-It assessments. The Solve It assessments are tasks designed at the DevTech Research Group to capture student's learning of programming concepts, from basic sequencing up through conditional branching (conditional branches are sequences that only occur under certain conditions, like when a distance sensor is activated by a robot coming near a wall) (Strawhacker et al. 2013). The assessments require children to listen to stories about a robot based on a familiar context and then attempt to create the program using CHERP paper icons. For example, one story was about the bus from the children's song "Wheels on the Bus," since children already knew the song and had a reference point for how that robot (bus) should behave (see Fig. 3).

Assessments were administered at a mid-point in the intervention and shortly after it ended, so that children's programming comprehension would be assessed after all of the isolated condition lessons, and again after the hybrid condition lessons.

Data was collected from each student in the form of five Solve Its on each assessment day (ten per child total). The Solve Its tested: simple sequencing, advanced sequencing, simple repeat loops, advanced repeat loops, and program construction. This analysis will focus on advanced sequencing and simple and advanced repeat loops.

Sequencing tasks consisted of a list of instructions for children to place in correct order. After hearing a verbal description of the program, children were able to sort through paper cut-outs of the CHERP language icons and organize them into the program that they had just heard (see Fig. 4). Sequencing is a core skill for young children, important in story-telling, planning a day's activities, and ordering numbers, and is considered a foundational skill in early mathematics and pattern comprehension (Kazakoff et al. 2013; Clements and Samara 2009).

Table 2 Timeline of curriculum conditions

	Lesson												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Condition													
Lesson content	–	Programming: sequencing			–	Programming: report loops			Mid-test	Programming: creative goal			Post-test
Class A	–	TUI			–	TUI			–	HUI			–
Class B	–	GUI			–	GUI			–	HUI			–
Class C	–	HUI			–	HUI			–	HUI			–

Story   Components:
- Context: *"Do you know the song, Wheels on the Bus? [Sing song with class]. I want to make a robot that is a bus, and I want its wheels to spin around four times, just like in the song."*
- Instructions: *"How would I do that? First, I want the robot bus to turn on. Next, I want the robot to spin its wheels, and to keep doing it four times. Then, I want the bus to stop spinning its wheels, and last, to turn off."*
- Challenge: *"Can you imagine the program my robot needs? Are you ready to try to make the program for my robot?"*

Fig. 3 Sample script for administering "Wheels on the Bus" Solve It task. Lead researchers administered all Solve It tasks, with aid from research assistants and regular classroom teachers

Fig. 4 Sample student response to advanced sequencing Solve It

The repeat loop tasks are similar to the sequencing tasks, except that instead of simply ordering actions, students are required to incorporate control flow blocks into the program (see Fig. 5). These control blocks are similar to the begin and end blocks, in that they begin and end a sub-section of a program to repeat. For both repeat-loop and sequencing Solve Its, the difference between Simple and Advanced tasks is the number of blocks required to correctly sequence the program. The advanced repeat loop questions also required children to separate repeating instructions from non-repeating instructions, a patterning task which is cognitively more challenging than simply identifying and ordering the instructions (Kazakoff et al. 2013; Clements and Samara 2009).

Solve Its were scored using a task-specific, analytical rubric developed by the DevTech Research Group (Strawhacker et al. 2013; Moskal 2000). Each question received a total score based on separate criteria, including placement of begin and end blocks and relative order of action blocks. The scoring rubric was developed after a pilot assessment was administered, to identify incorrect answer patterns that could demonstrate developmental level rather than programming comprehension. For example, many children reversed the order of their program so that it was read right to left. Rather than mark these responses incorrect, researchers observed that in all the cases when this occurred, those children worked right to left, indicating that their logical sequencing ability was actually separate from their directional sense. This led to adjustment of the metric to assume that in cases when a student's response would earn more correct points if reversed, they should be scored that way. Inter-scorer reliability tests showed precise agreement (two items; $K = 0.902$, $p < 0.001$).

Student responses to the Solve It tasks were scored on a 0-6 rubric, with two 3-point subscales targeting concepts of "control flow" and "action" sequencing. Research assistants who had not worked on developing the Solve It assessments used responses to develop the scoring rubric. For example, sequencing rubrics allow for more point variation

Fig. 5 Sample student response to advanced repeat-loop Solve It

depending on how the actions were ordered, while repeat loop rubrics focus more on correct loop structure in a program (see Figs. 6, 7 and 8).

In addition to the Solve It tasks, qualitative data was collected in the form of video footage of the children working during lesson time. Video data was collected of the children during all small group activity time for seven class sessions (one camera per small group), starting on class 4. Classes 1–5 were not recorded because on these days children were exploring the robotic equipment, and programming was still being introduced. Footage was not collected on buffer days or during assessments. This left seven lesson days per class (105 total hours of footage) that we recorded for later analysis.

Children's classwork artifacts, including written and drawn programming plans, untested CHERP programs, and videos of finished robotic projects, were also collected in order to supplement and illustrate themes derived from data analysis. Research assistants in teaching roles also recorded informal journal entries and subjective notes to illuminate what happened in the classrooms. Finally, a select few groups were chosen to undergo more rigorous qualitative examination. These case-study groups will not be discussed thoroughly here, although some trends and themes from their coded transcripts will be mentioned in the Discussion portion of the paper.

Data analysis

To answer our first research question, "Does the interface style of the language affect a child's comprehension of programming concepts?" Solve It assessment scores were compared across classrooms. In the TUI and GUI condition groups, the Solve-It scores provide a quantitative baseline to compare each child's performance before and after the introduction of the new UI experience. The HUI condition class serves as a third comparison group.

Our second research question, "When do children switch between interfaces, and are how are they using them?" required analysis of how the children spent their time programming. Video footage was coded using time interval sampling to mark what type of programming activity a child was doing at any given time: building a program with blocks, building a program on the computer, compiling and testing a program, rebuilding their robot, engaging in a non-lesson related task, etc. Video sections were sampled such that the first minute of every 5-min segment was coded (Repp et al. 1976). Events when children were actively using the programming interfaces were marked as rich events, and coders were able to view those instances for longer than 1 min to see how an event played out (Friedlander et al. 1972).

Solve It scores were analyzed to compare means within and across classrooms for statistically significant trends. These data were then compared using linear regressions with trends in the time-sampled interface-use in order to triangulate the results.

Fig. 6 Correctly sequenced response to the "Wheels on the Bus" Solve It task. Begin and end blocks are correctly located (3 out of 3 points). Repeat and end-repeat blocks and number parameter correctly surround spin block (3 out of 3 points). Total: 6 out of 6 points

Fig. 7 Incorrectly sequenced response to the "Wheels on the Bus" Solve It task. Begin and end blocks are correctly located (3 out of 3 points). Number parameter is correctly placed on repeat block, and repeat block is correctly located before end repeat block. However, spin block is incorrectly placed outside of repeat loop. This rendered the repeat loop non-functional (1 out of 3 points). Total: 4 out of 6 points

Fig. 8 Incorrectly sequenced response to the "Wheels on the Bus" Solve It task. Begin and end blocks are incorrectly switched (0 out of 3 points), number parameter is correctly placed to side of repeat block, end repeat block is incorrectly located before repeat block, and spin block is incorrectly placed outside of repeat loop (0 out of 3 points). This response would score a 0 out of 6 points, except for the amendment in the scoring rubric to account for children working right-to-left. Because of this, the response is scored as if it had been written backward, and scores a total of 2 out of 6 points

Although presenting qualitative case-studies is beyond the scope of this paper, video footage taken of two children from the TUI classroom and three from the HUI classroom was transcribed and read by two researchers for emergent themes in the data (Charmaz 2006). Transcripts were scrutinized further to arrive at more granular codes within each theme, and a codebook was created from the codes.

Analysis of Solve It scores

Due to the small size of this sample (N = 35) and the nature of working with children in a public school setting with mixed developmental levels (and in some cases, confidential learning disorders), the results presented here should be interpreted with caution. Because

of the small data set, statistical inferences were slightly harder to make. For example, the data did not meet all statistical assumptions for performing a one-way ANOVA. The data failed the assumption of normality, although it is likely that this assumption would be met had a larger sample size been used. Normal transformations were attempted (e.g. logarithmic transformation as well as square-root transformation), however this still did not yield normal distributions. Although ANOVA is very robust to "violations" of normality, non-parametric tests were performed. In this case, the Kruskal–Wallis test was used to determine if the distribution was the same across classroom groups and Wilcoxin Signed Rank test to compare the median of pre and post scores within classrooms. Both parametric and non-parametric tests led to the same conclusions. Therefore, the ANOVA and repeated measures tests will be discussed in these results.

Results

Time interval sampling of the video footage allowed the identification of UI trends over time. The graphs below represent the cumulative events in each classroom of children using computers, using blocks, and testing their program on a robot (see Figs. 9, 10 and 11). The number of interactions per day was normalized to account for the number of children in each classroom. Testing was included as a coding category because it represents the time that children spend observing a robot act out the program they made. This kind of iterative feedback and modification is an integral step in the programming process. Instances when children were programming, or "interaction events," were defined as any event when a child was engaged with a computer or with blocks. The dotted line on the TUI graph represents the day that the Tangible condition classroom was introduced to the computers. Similarly for the GUI classroom, the dotted line indicates when blocks were introduced.

For all graphs below, "Day" refers to programming-only days when video footage was collected. Days 1-5 are not included, since these days were primarily for introduction of programming concepts. Days 6–8 represent the Unit 2 of the curriculum, the last 3 days during which children worked in the isolated condition. Day 9 was an assessment day, followed by free exploration of programming tools. On days 10–12, Unit 3 was introduced and TUI and GUI children were exposed to new programming styles alongside their familiar ones. The HUI group worked with both styles for the duration of the curriculum.

Summary of Solve It scores

This analysis focuses on three of the five Solve-It tasks that all of the Kindergarten students completed during testing. The first task was a sequencing challenge that was relatively easy even for our pre-Kindergarten pilot group. This task was intentionally used as a warm-up exercise the Kindergarteners in this study, and allowed them to experience a Solve It task once before being assessed. The fifth task was a more challenging repeat-loop challenge and focused on block selection rather than action sequencing. This Solve It will require more thorough analysis and will be investigated in later work.

Each Solve It score was composed of two factors: a score for the begin and end block sequencing, and a score for the action block sequencing. Begin and end scores (0–3 points) and action scores (0-3 points) were added to get a total score for each question.

Table 3 demonstrates the mean begin and end scores for each Solve It task by classroom. Average scores on the "Begin and End" block criteria fell in the range of 2.43–3 out
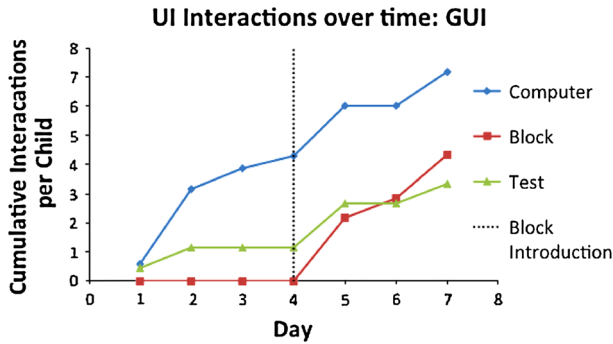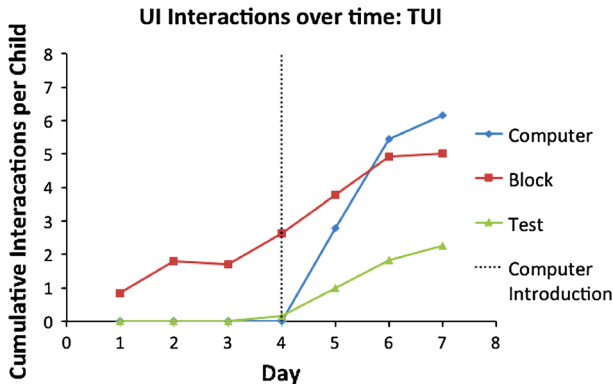
Fig. 9 Interface events in the GUI-condition classroom. This graph represents the cumulative frequency of programming interfaces used by children in the graphical condition group. Block use did not begin until the ninth programming class because that was midpoint when tangible-style programming was introduced. After blocks were introduced, computer use remained high, while block use and events of children testing and observing their robots acting out programs increased

Fig. 10 Interface events in the TUI-condition classroom. This graph represents the cumulative frequency of programming interfaces used by children in the tangible condition group. Computer use did not begin until the ninth programming class because that was midpoint when graphical interface programming was introduced. Similarly, students were not able to test their programs by watching robots act them out without adult assistance, because teachers were required to handle the computer interfaces at testing areas. After computers were introduced, computer use became the most popular programming interface, although block use remained high as well

of a max of 3 points in the mid-test and 2.64–3 points in the post. The lack of variation in mid- and post-test scores could be explained by a ceiling effect in the scoring rubric. It may also suggest that concept of begin and end blocks was already accessible to the students, so there was not much room for improvement. Most of the students spontaneously referred to the "go" and "stop" blocks, demonstrating their intuitive understanding of the relative sequence of start and stop instructions. Therefore, one might expect a ceiling effect on these scores.

Table 4 showed the mean scores for the action block sequencing of each Solve It task by classroom. Average scores on action blocks showed slightly more variation, with most scores falling in the range of 1.29–2.17 out of 3 points in the mid-test and 1.29–2.5 points
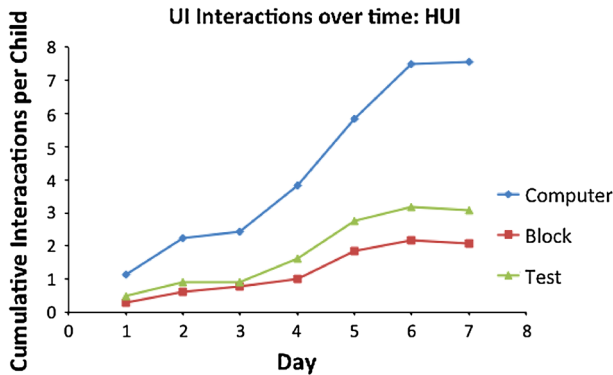
UI Interactions over time: HUI

Fig. 11 Interface events in the HUI-condition classroom. This graph represents the cumulative frequency of programming interfaces used by children in the hybrid condition group. Computer use remained high throughout the intervention. Testing programs by running them on robots was more frequent than children using the blocks to build programs

in the post-test. Of these results, the most pronounced change over time occurred in the TUI classroom on the easy repeats task. For this task, students in the Tangible classroom earned the lowest average scores across all three classes in the mid-test, and went on to earn the highest average scores in the post-test.

The means for total scores on each Solve It task are shown in Table 5. Average total scores fell in the range of 4.07–5.17 out of 6 points in the mid-test and 4.29–5.5 on the post-test. Generally, total scores improved from mid- to post-test across all tasks and classrooms, with the only exception being the HUI group's scores on the Hard Sequencing task. For this task, HUI classroom students showed no change over time in average scores, although the larger standard deviation of the post-test indicates that their scores varied more over time.

There are some interesting patterns in the Solve It responses. However, given the small sample size and the lack of many significant findings, these results should be interpreted with caution.

One interesting finding is that in every Solve It task, the HUI group earned the lowest average total scores across all classrooms for both the mid- and post-tests. A closer look at these scores shows that the HUI group's begin and end score averages were comparable to those in the other classrooms, meaning that the disparity in Total scores must be related to action block scores. Indeed, the HUI students earned the lowest average Action scores of all three classrooms for both the mid- and post-tests of every Solve It task. While this finding is not statistically significant, it does suggest that there may be a learning benefit to introducing programming with a single interface style first, rather than two at once.

The trends in average Total scores of the GUI and TUI groups were also interesting. Students in the GUI group outperformed TUI classroom students in mid-tests on all three tasks, but the post-test results showed more variation. On the Hard Sequencing task, the GUI group scored the highest average Total scores, while the TUI group earned the highest Total scores on both repeat tasks. Repeat loops and other complex, nested patterns are generally considered to be conceptually challenging concepts that build off of the concept basic mathematical sequences. For example, Massachusetts curriculum standards focus on "number order and basic operations" in Kindergarten, but do not introduce "using patterns to solve problems" until the 3rd grade (Massachusetts Department of Elementary and

Table 3 "Begin and End" scores by classroom for mid- and post-tests

Begin/end	Hard sequencing task			Easy repeats task			Hard repeats task		
Classroom	N	Mean	SD	N	Mean	SD	N	Mean	SD
TUI									
Mid	14	2.86	0.535	14	2.86	0.535	14	2.86	0.535
Post	13	2.85	0.555	12	3	0	12	3	0
GUI									
Mid	6	3	0	6	3	0	6	2.83	0.408
Post	7	3	0	7	2.86	0.378	7	2.71	0.756
HUI									
Mid	14	2.86	2.86	14	2.43	0.852	14	2.57	0.756
Post	14	2.64	2.64	14	3	0	14	3	0

Table 4 "Action" scores by classroom for mid- and post-tests

Totals	Hard sequencing task			Easy repeats task			Hard repeats task		
Classroom	N	Mean	SD	N	Mean	SD	N	Mean	SD
TUI									
Mid	14	1.86	1.292	14	1.29	1.204	14	1.64	0.633
Post	13	1.92	1.038	12	2.5	0.905	12	2.08	0.669
GUI									
Mid	6	2.17	0.983	6	2	1.549	6	1.67	0.816
Post	7	2.43	0.976	7	2.29	1.254	7	1.71	0.756
HUI									
Mid	14	1.64	1.082	14	1.64	1.336	14	1.57	0.646
Post	14	1.86	1.231	14	2	1.038	14	1.29	0.914

Table 5 Total scores by classroom for mid- and post-tests

Totals	Hard sequencing task			Easy repeats task			Hard repeats task		
Classroom	N	Mean	SD	N	Mean	SD	N	Mean	SD
TUI									
Mid	14	4.71	1.49	14	4.14	1.46	14	4.5	0.941
Post	13	4.77	1.301	12	5.5	0.905	12	5.08	0.669
GUI									
Mid	6	5.17	0.983	6	5	1.549	6	4.5	1.049
Post	7	5.43	0.976	7	5.14	1.574	7	4.43	0.976
HUI									
Mid	14	4.5	1.345	14	4.07	2.056	14	4.14	1.231
Post	14	4.5	1.951	14	5	1.038	14	4.29	0.914

Total scores were derived by summing the "Begin and End" and "Action" scores for each Solve It task

Secondary Education 2011). It is interesting to find in this study that better understanding of the concept of sequencing does not predict better understanding of more complex sequences involving repeat loops, and in fact TUI Kindergarteners showed slightly stronger understanding of repeat loops than linear sequences. This finding may call into question the assumption that linear sequencing is a foundational skill for understanding looping functions.

In order to answer our research questions, analysis was conducted to determine whether any significant patterns emerged in children's performance on the Solve It assessments. Univariate ANOVA analysis was performed to determine if there were differences in means for mid-test scores amongst the classrooms (see Fig. 12). There were no statistically significant differences in performance, so all classrooms earned Solve It scores in roughly the same range. This suggests that the interaction style (tangible, graphical, or hybrid) had no significant effect on children's programming comprehension during the first two units of the curriculum.

Univariate ANOVA analysis was also performed to determine if there were differences in means for post-test scores amongst the classrooms (see Fig. 13). There was one statistically significant difference between the TUI and HUI classrooms on the hard repeat task ($F = 3.657$, $p = 0.038338$). This finding shows that the TUI group significantly outperformed the HUI group on this hard repeat task. For all other tasks, there were no differences in post-test scores between classrooms.

Repeated measures ANOVA analysis was run to determine if there were significant differences across classroom from mid- to post-test (see Fig. 14). Average Solve It scores did not vary significantly across tasks when all classrooms were considered. When variations within classrooms were considered, a repeated measures ANOVA for the TUI classroom did show a significant increase in scores from the mid-test to the post-test on two Solve It tasks: the easy repeat task ($F = 13.156$, $p = 0.003069$) and the hard repeat task ($F = 6.656$, $p = 0.02286$) (see Table 6). This could mean that for students initially exposed to tangible programming, the later addition of a graphical UI may offer important advantages for concept mastery. However, since there was little variation in final scores among the classrooms, it is difficult to say whether this advantage signifies better overall understanding, or just a steeper learning curve (i.e. a clearer "aha! moment") for students learning to program with CHERP.

The HUI group showed no statistically significant change across all questions, which might be expected since their experimental condition did not change throughout the study. Interestingly, although the TUI group showed significant gains in understanding after the introduction of the computer interface, the GUI group did not show any statistically significant changes after being exposed to the wooden CHERP blocks. Although the TUI group did show significant improvement from mid- to post-tests within the class, they did not perform significantly worse on the mid-test or better on the post-test than the other classes.

Table 7 shows the difference in total mean scores between each classroom, highlighting differences that indicate a significant variation in performance between classes. The mean scores for the hard repeat (total) Post Test, the TUI group significantly outperformed the HUI group ($p = 0.016*$). TUI also outperformed the GUI group on this question, although the difference in means was not statistically significant ($p = 0.096$). However, the difference between GUI and TUI was large enough that they may have shown to be significantly different in a larger study with more student scores. This is especially interesting since the difference between the HUI and GUI classrooms were marginal, almost approaching no difference at all ($p = 0.965$).

Data collected from Solve It assessments were triangulated with data about the frequency of UI interactions, to determine if there were any significant relationships. A closer
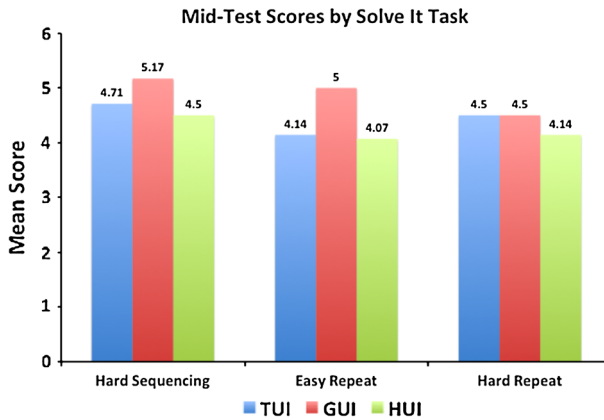
Mid-Test Scores by Solve It Task

Fig. 12 Mid-test comparisons across the three classrooms for each Solve It task
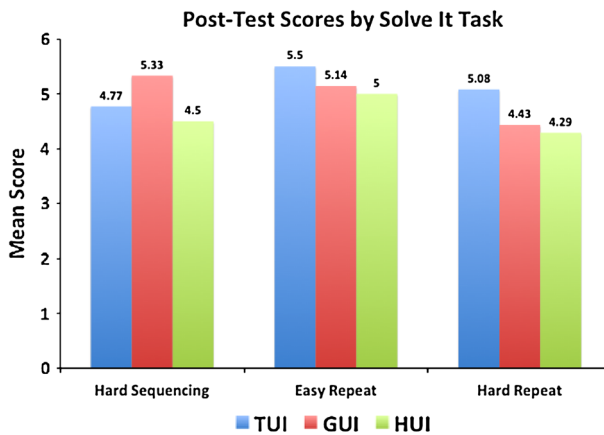
Post-Test Scores by Solve It Task

Fig. 13 Post-test comparisons across the three classrooms for each Solve It task

look at the UI interactions reveals that when children had access to both blocks and computers, the graphical UI was consistently more popular and more often used than the block interface. This could possibly be due to a novelty effect of having a tool in the classroom that children would commonly associate with adults. Although no data was collected about this phenomenon, many research assistants reported that when computers were offered as a programming option in class, they repeatedly heard the children telling each other stories about their parents and family members using computers at home.

Linear regressions were run to determine whether the TUI group's significantly improved programming comprehension was related to their ability to begin testing programs immediately after creating them. This seemed plausible, considering that in previous units, they had to wait for up to 15 min to test out their programs. They may have already forgotten what they created in that time, so recognizing errors during testing could have been much harder for TUI students. All results were non-significant. Based on these findings, it cannot be concluded that the amount of time spent interacting with different interface types had an affect on Solve It scores.
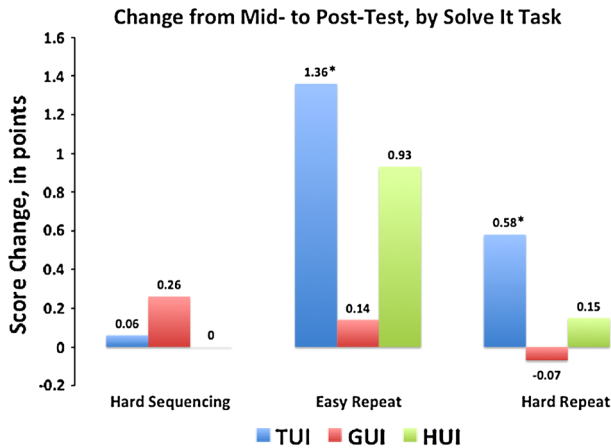
Fig. 14 Change in mid- to post-test scores within each classroom, by question type. (* indicates statistically significant change from pre- to post-test.)

Table 6 *p* values by classroom

Classroom	*p* values by task types		
	Hard sequencing	Easy repeat	Hard repeat
TUI	0.593	0.003*	0.023*
GUI	0.203	0.235	0.572
HUI	0.973	0.127	0.655

Values represent significance levels for change in each classroom from mid- to post-test. The TUI classroom showed statistically significant changes from mid- to post-test

* Statistically significant at probability of 0.05

Table 7 Univariate ANOVA multiple comparisons table for dependent variable hard repeat post, with Tukey post hoc test

	TUI	GUI	HUI
TUI	–	0.096	0.016*
GUI	0.096	–	0.965
HUI	0.016*	0.965	–

Discussion

Overall analysis of the Solve Its results reveals little difference in scores across the three interface conditions. However, different condition classrooms showed variations in concept mastery by the end of the intervention, with the TUI group outperforming the GUI and HUI classroom on tasks about repeat loops, and the GUI classroom scoring highest on the Sequencing task. All three classrooms showed slight improvement in mean scores from mid- to post-test for most tasks, with the TUI group showing statistically significant improvement on both repeat loop tasks. The only exception is the HUI classroom, whose mean scores dropped in the post-test hard repeat loop task, although it is difficult to draw

conclusions from this finding because the change was statistically insignificant. This is most likely due to the fact that the sample size of each class was very small.

The GUI group's scores remained the most consistent across mid- and post-tests, and consistently outperformed the HUI group on all tasks. The TUI group scored slightly below the GUI group on the mid-tests, but slightly above GUI on the post-test repeat loop tasks. Based on the mean total Solve It scores, the HUI group underperformed the other classes in all three Solve It tasks. However, because these results were not statistically significant, it cannot be concluded whether this trend that would be more robust in a larger study.

The few statistically significant results suggest that something interesting happened with the TUI group during the third unit of the curriculum. This classroom showed significant improvement from the mid- to post-test on tasks involving repeat loops, and significant improvement in the hard repeat task post-test when compared to the HUI classroom.

Triangulating the data: UI interactions and Solve It scores

During the last phase of the curriculum, when students had access to both programming interfaces, block use was higher in the TUI and GUI classrooms than in the HUI classroom. This could be caused by a novelty effect in the GUI classroom, which had only used computers before, and heightened familiarity in the TUI group. Testing programs also remained slightly lower in the TUI group in the last curriculum phase, even though they frequently used computers (which they would need for testing). Finally, the TUI classroom was the only group that used blocks more than computers when given the option to use both, and they also had the most days when block and computer use were almost evenly-split.

The lack of significant relationship found between time spent testing programs and performance on Solve It tasks means that we cannot conclude whether seeing robots running programs contributed to students' understanding of the block commands. Of the three classrooms, the TUI group still spent the least amount of time testing programs, implying that the rapid improvement was not related to the ability to test programs.

One explanation for the lack of significant relationships between testing programs and program comprehension is that the TUI group was primed to plan their programs for a longer time before testing. The limited access to testing stations in this classroom meant that they had to wait in line before they could watch their robots executing a program. By having to wait so long between building and testing a program, perhaps they spent more time wondering what the robot would eventually do, and were ready to understand errors when they saw them. Alternatively, they could have spent their time in line observing other students testing their programs. Since the time sampling only accounted for students testing and observing their own robots, it is possible that observing peer programs is a confounding variable. A third explanation is that perhaps TUI students only needed a relatively small amount of time to test programs because they had been primed so well by using block programming that it forced them to slow down and internalize the concepts. More research is needed to confirm or reject any of these hypotheses.

This might also explain why the HUI group slightly underperformed throughout the intervention, even though they had access to all of the materials from the start. Perhaps the variety of programming interfaces was overwhelming. If a student faced a difficult programming or collaboration issue, he could easily turn from building a block program to a computer one without retaining his original programming goal or working through the underlying concepts.

Focus on TUI: Why the improvement?

The TUI group showed the most dramatic change in performance from mid- to post-test Solve It tasks. Between the two tests, the TUI classroom was exposed to the third unit of the robotics intervention, which involved three changes: a digital interface, faster and more frequent testing, and child-selected programming goals to guide lessons.

The first change was the introduction of a computer interface for programming, alongside the already familiar wooden blocks. This meant that children could now use an all-digital interface for constructing programs, in addition to the tangible block interface with which they were already familiar. This new interface may have helped crystallize ideas that the tangible language had already fostered. For example, a program with a repeat instruction but no end repeat instruction is possible to create with the wooden blocks, but it will not compile on the computer. The computer interface will warn the user when they try to compile such a program that "Every Repeat needs an End-Repeat" (see Fig. 15). This could account for the steep increase in comprehension of repeat loops.

Second, they were able to test programs immediately at their seats. The CHERP programming environment requires a robot to be physically plugged into a computer in order to run. Because of the nature of the study, the children in the TUI classroom could not be exposed to the computer version of the software, so for the first two sessions they waited in line at "program testing stations" run by teachers and research assistants. There were unforeseen effects to this approach. Having computers present but not available sparked the students' curiosity about the graphical software, so they had already formed hypotheses about what would be on the computers and how to use them. The testing stations also were limited in number so children were usually watching other students' programs run and looking at their programs. Informal observation and collaboration may have played a role in the development of the TUI children's ideas about programming. With the introduction of a computer for each group's robot, the TUI children were allowed to test programs immediately upon creating them, and also to test them repeatedly and make minor changes, which they often did.

The third change for the TUI group in the last curriculum session was the introduction of creative programming. This portion of the curriculum allowed children to come up with a personal goal for their program, which theoretically could promote more exploration and subsequent mastery of programming concepts. However, there is no conclusive evidence that creative programming had an effect on programming comprehension. Because all three classrooms were exposed to this unit, it might be expected that they would all show changes in Solve It scores. In fact, there was slight overall improvement, but without a larger sample to verify trends there is no way of knowing whether the improvement was influenced by the creative programming unit, the experimental classroom conditions, or something else entirely. It also cannot be determined if the improvement represented the natural learning trajectory of the children as they gained more experience using the CHERP programming language.

Classroom trends

There were a few trends that were salient across all three classrooms. Because of the limited space in the classrooms, the materials that were not being used were usually stored in plain sight in a corner of the room. The children saw the other materials and were often very curious about when they would be able to use them. The TUI group was eager to work with the computers, and GUI group was much more curious about blocks than the other

Fig. 15 CHERP graphical interface warning. This computer warning occurs when a repeat loop is syntactically incorrect and will not compile. The message prompts the user to include the correct block

two classrooms. This was despite explicit instruction and even overt attempts to restrict access to the non-introduced interfaces during the early curriculum units.

It is possible that this built-up excitement about the materials affected the level of interface engagement, however the extra materials that were visibly stored were not just unused interfaces, but extras of all materials that were already freely available in the classroom (i.e. Lego bricks, computers in the GUI classroom, CHERP blocks in the TUI group, craft materials for decorating the robots, etc.). Since the children in all three classrooms were curious about extra materials, even when they already had the same materials at their workspaces, it is unlikely that this factor influenced the children's choices about which materials to program with.

Another common theme was the way that children talked about their animals and programs. In all classes, students were able to switch between calling their project a robot and an animal. Although it appeared to researchers that most children understood that the robot was a non-living, symbolic representation of an animal, it is difficult to know because children regularly assigned anthropomorphic goals to their "robot," or alternatively described programming actions that their "tiger" would execute (see Figs. 16 and 17).

The fact that children understood that their robots abstractly represented animals was interesting. Since it is traditionally believed that children in Kindergarten are just
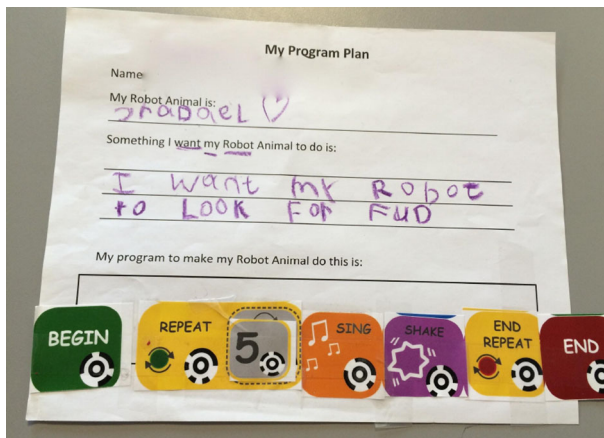

Fig. 16 Program planning sheet. Text reads: "My Program Plan. Animal: Crocodile. Program: I want my robot to look for food"
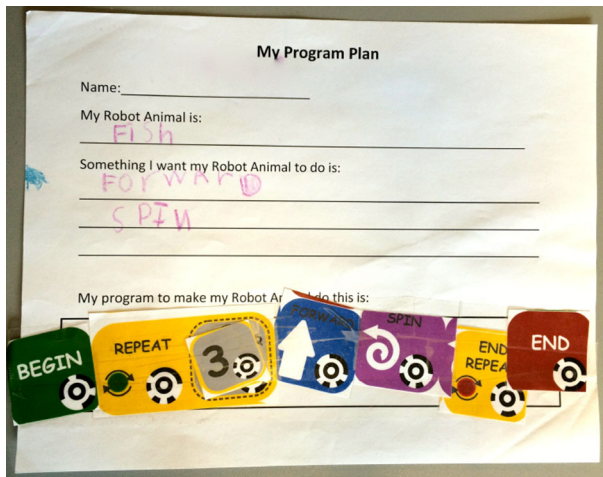
Fig. 17 Program planning sheet. Text reads: "My Program Plan. Animal: Fish. Program: Forward, Spin"

beginning to experiment with abstract and symbolic thought, this was expected to be more a barrier to understanding (Piaget 1952). However, classroom artifacts made by the children demonstrate that they had no trouble switching between talking about the robot as a physical, inanimate object and also as a living animal.

This type of mental abstraction may explain how the children were able to understand the concept of repeat loops. Essentially, a repeat loop in the CHERP language requires a programmer to remember the larger program that begins with begin and ends with end, and also a smaller program, that begins with repeat and ends with end-repeat. This sounds like a complex concept for a child of five, but in a way, repeat loops are simply an abstraction of a linear program. Despite the fact that this kind of symbolic reasoning was not expected until age seven by Piagetian standards, the students in all three classrooms were able to demonstrate some kind of understanding in class and on their Solve It assessments.

Limitations and future work

This research was a pilot study conducted with three classrooms in the same school setting. Although this narrowed variability in the children's school environment, there were several challenges posed by working within the school environment. The small sample size of all three classrooms makes it difficult to draw conclusions about the results. This is especially challenging since the GUI group is much smaller than the other classrooms. This is because, as explained in earlier sections, the GUI group was composed of the children aged 5 and up from a larger mixed-age Montessori classroom. This is simply due to the fact that the participating school site only had two age-graded Kindergarten classrooms. The other Kindergarten students were part of Montessori classes.

Since this is a pilot study, it cannot be concluded whether any of the differences shown in the GUI group are related to the different interface style of the computer, the learning style of the students accustomed to Montessori-style teaching, or the small size of the sample. In future work on programming interactions in Kindergarten, this study could be expanded to explore the patterns across larger groups of students, and with more uniform

school settings. A more even sample size of each interaction group would also help to highlight statistically significant differences in performances on programming assessments.

This study focused on the frequency and type of child-interface interaction, and child performance on an assessment of programming knowledge using the CHERP language. Although many of the research questions outlined in the beginning of this paper have been addressed, it is difficult to say what caused children to switch between different user interfaces and why. Since children's performance on Solve Its was not significantly correlated with the frequency of UI interactions, a closer look is needed at the qualitative experience of children using the programming language. Depth qualitative inquiry could be conducted in the form of qualitative coding of the collected video footage to examine what cognitive learning patterns are associated with successful programmers using the computer, block, and the hybrid interfaces. This study aimed to examine the differences in learning among Kindergarten students when the same programming language and robotics curriculum was introduced through tangible, graphical, and hybrid interface styles. Although the results are inconclusive, this pilot study shows that for Kindergarten students, using a tangible programming language may be related to enhanced comprehension of abstract concepts like repeat loops.

While this is an exciting finding, computer programming for young children is still a very recent concept. More investigation needs to be done on the learning patterns and challenges of kindergarteners working with tangible technologies, so that educators may provide the most beneficial educational materials and learning experiences for young children exploring programming concepts.

References

Ackermann, E. (1996). Perspective-taking and object construction. In Y. Kafai & M. Resnick (Eds.), *Constructionism in practice: Designing, thinking, and learning in a digital world* (pp. 107–123). Mahwah, NJ: Lawrence Erlbaum Associates.

Africano, D., Berg, S., Lindbergh, K., Lundholm, P., Nilbrink, F., & Persson, A. (2004). Designing tangible interfaces for children's collaboration. In *CHI'04: CHI'04 extended abstracts on Human factors in computing systems* (pp. 853–868). New York, NY: ACM.

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

Bers, M. U. (2010). The TangibleK robotics program: Applied computational thinking for young children. *Early Childhood Research and Practice, 12*(2). http://ecrp.uiuc.edu/v12n2/bers.html.

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

Bers, M., & Horn, M. (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.

Charmaz, K. (2006). *Constructing grounded theory: A practical guide through qualitative analysis*. London: Sage Publications.

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.

Clements, D. H. & Samara, J. (2009). Effects of a preschool mathematics curriculum: Summative research on the building blocks project. *Journal for Research in Mathematics Education, 38*(2), 136–163. http://gse.buffalo.edu/fas/clements/files/clements_bb_jrme.pdf.

Creswell, J. W. (2014). *Research and design: Qualitative, quantitative, and mixed methods approaches* (4th ed.). Thousand Oaks, CA: SAGE Publications Inc.

Frei, P., Su, V., Mikhak, B., & Ishii, H. (2000). *Curlybot:* Designing a new class of computational toys. T. Turner & G. Szwillus, Conference Editors. In *SIGCHI Conference on Human Factors in Computing Systems,* New York, NY: ACM Press.

Friedlander, B. Z., Jacobs, A. C., Davis, B. B., & Wetstone, H. S. (1972). Time-sampling analysis of infants' natural language environments in the home. *Child Development, 43*(3), 730–740.

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

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.

Kazakoff, E. R., Sullivan, A. 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), 245–255.

Kim, M. J., & Maher, M. L. (2006). *Comparison of designers using a tangible user interface and a graphical user interface and the impact on spatial cognition* (Doctoral dissertation). Key Centre of Design Computing and Cognition, University of Sydney, Australia.

Kostelnik, M. J., & Grady, M. L. (2009). *Getting it right from the start: The principal's guide to early childhood education*. Thousand Oaks, CA: Corwin.

Manches, A., & Price, S. (2011). Designing learning representations around physical manipulation: Hands and objects. *In 10th international conference on interaction design and children,* Ann Arbor, MI, NY: ACM Press.

Marshall, P. (2007). Do tangible interfaces enhance learning? In *First international conference on tangible and embodied interaction*, Baton Rouge, LA.

Massachusetts Department of Elementary and Secondary Education (MA DOE) (2011). *Massachusetts Curriculum Framework for Mathematics, Grades Pre-Kindergarten to 12*. Malden, MA: Massachusetts Department of Elementary and Secondary Education. http://www.doe.mass.edu/frameworks/math/0311.pdf.

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.

Moskal, B. M. (2000). Scoring rubrics: What, when and how? *Practical Assessment, Research & Evaluation, 7*(3). http://PAREonline.net/getvn.asp?v=7&n=3.

Piaget, J. (1952). *The origins of intelligence in children*. New York: International Universities Press.

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

Ploderer, B. (2005). *Tangible User Interfaces: Potentials Inherent in Tangible User Interfaces for Simplified Handling of Computer Applications (Master's thesis)*. Graz, Austria: University of Applied Sciences FH JOANNEUM.

Quarles, J., Lampotang, S., Fischler, I., Fishwick, P., & Lok, B. (2008). Tangible user interfaces compensate for low spatial cognition. In *IEEE symposium on 3D user interfaces*, Reno, Nevada.

Raffle, H. S., Parkes, A. J., & Ishii, H. (2004). Topobo: a constructive assembly system with kinetic memory. In *SIGCHI conference on human factors in computing systems*. Vienna, Austria: ACM Press.

Repp, A. C., Roberts, D. M., Slack, D. J., Repp, C. F., & Berkler, M. S. (1976). A comparison of frequency, interval, and time-sampling methods of data collection. *Journal of Applied Behavioral Analysis, 9*(4), 501–508.

Resnick, M., Martin, F., Berg, R., Borovoy, R., Colella, V., & Kramer, K. et al. (1998). Digital manipulatives: New toys to think with. M. E. Atwood, C. Karat, A. Lund, J. Coutaz, & J. Karat, Conference Editors. In *SIGCHI conference on human factors in computing systems*, New York: ACM Press.

Sehyun, A. (2006). *Hybrid user interfaces: design guidelines and implementation examples* (Doctoral dissertation). Retrieved from DSpace@MIT. http://hdl.handle.net/1721.1/34382.

Strawhacker, A., Sullivan, A., & Bers, M. U. (2013). TUI, GUI, HUI: Is a bimodal interface truly worth the sum of its parts? In *Proceedings of the 12th international conference on interaction design and children (IDC '13)* (pp. 309–312). New York, NY: ACM Press.

Sullivan, A., Kazakoff, E. R., & Bers, M. U. (2013). The wheels on the bot go round and round: Robotics curriculum in pre-Kindergarten. *Journal of Information Technology Education: Innovations in Practice, 12*, 203–219. http://www.jite.org/documents/Vol12/JITEv12IIPp203-219Sullivan1257.pdf.

Suzuki, H., & Kato, H. (1995). Interaction-level support for collaborative learning: AlgoBlock—an open programming language. John L. Schnase and Edward L. Cunnius, Conference Editors. In *First international conference on computer support for collaborative learning (CSCL '95)* L. Erlbaum Associates Inc., Hillsdale, NJ.

Tsong, C. K., Chong, T. S., & Sumsudin, Z. (2012). Tangible multimedia: A case study for bringing tangibility into multimedia learning. *The Turkish Online Journal of Educational Technology (TOJET), 11*(4), 442–450. http://www.tojet.net/articles/v11i4/11445.pdf.

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.

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

Xie, L., Antle, A. N., & Motamedi, N. (2008). Are tangibles more fun? Comparing children's enjoyment and engagement using physical, graphical, and tangible user interfaces. *Second International Conference on Tangible and Embedded Interaction*. Bonn, Germany, Feb 18–20.

Xu, D. (2005) Tangible user interface for children—an overview. In *Sixth Conference in the Department of Computing, University of Central Lancashire*, Preston, UK.

Zuckerman, O., Arida, S., & Resnick, M. (2005). Extending Tangible Interfaces for Education: Digital Montessori-inspired Manipulatives. In *ACM CHI 2005 Conference on Human Factors in Computing Systems,* Oregon, US.

Zuckerman O., & Resnick M. (2003). System blocks: A physical interface for system dynamics learning. In *The 21st International System Dynamics Conference.*