6-11-2011

Tangible Interaction and Learning: The Case for a Hybrid Approach

Michael S. Horn
Northwestern University

R. Jordan Crouser
Tufts University, jcrouser@smith.edu

Marina U. Bers
Tufts University

Pers Ubiquit Comput
DOI 10.1007/s00779-011-0404-2

ORIGINAL ARTICLE

Tangible interaction and learning: the case for a hybrid approach

Michael S. Horn · R. Jordan Crouser ·
Marina U. Bers

Received: 11 October 2009 / Accepted: 27 January 2010

Abstract Research involving tangible interaction and children has often focused on how tangibles might support or improve learning compared to more traditional methods. In this paper, we review three of our research studies involving tangible computer programming that have addressed this question in a variety of learning environments with a diverse population of children. Through these studies, we identify situations in which tangible interaction seems to offer advantages for learning; however, we have also identify situations in which tangible interaction proves less useful and an alternative interaction style provides a more appropriate medium for learning. Thus, we advocate for a hybrid approach—one that offers teachers and learners the flexibility to select the most appropriate interaction style to meet the needs of a specific situation.

Keywords Tangible interaction · TUIs · Hybrid tangible interface · Computer programming · Education · Children · Robotics

M. S. Horn (✉)
Computer Science and the Learning Sciences,
Northwestern University, Evanston, IL 60208, USA
e-mail: michael-horn@northwestern.edu

R. J. Crouser · M. U. Bers
Eliot-Pearson Department of Child Development,
Tufts University, Medford, MA 02155, USA
e-mail: ryan.crouser@tufts.edu
URL: http://ase.tufts.edu/DevTech/

M. U. Bers
e-mail: marina.bers@tufts.edu

1 Introduction

Research involving tangible interaction and children has often focused on how tangibles might support or improve learning compared to more traditional methods [1, 7, 10, 14, 15]. This research has included an emphasis on the production of frameworks that seek to guide designers toward promising activities and learning situations for tangible interaction. Our own research involving computer programming has compared tangible and graphical interaction in a variety of learning environments with a diverse population of children [3, 5, 10]. Through these studies, we have identified situations in which tangible interaction seems to offer advantages for learning; however, we have also identified situations in which tangible interaction is less useful and an alternative interaction style provides a more appropriate medium for learning. These situations are not always clear-cut. We cannot, for example, say that tangible interaction is better for young children and graphical interaction for older. Rather, the distinctions are subtler. Tangibles might be better for certain situations or for certain children in certain phases of the learning process. In this article, we advocate for a hybrid approach—one that offers users the flexibility to select the most appropriate interaction style for a given context and the ability to fluidly change from one interaction style to another.

Our position on hybrid tangible interfaces is based on analysis of data from research conducted in classrooms, after-school settings, and museums. In this paper, we will review findings from three studies that have compared the use of tangible and mouse-based interaction. Each study involved the use of *Tern*, a tangible programming language designed to allow students to control educational robots by constructing computer programs out of a collection of wooden blocks (Fig. 1) [9]. The first two studies compared

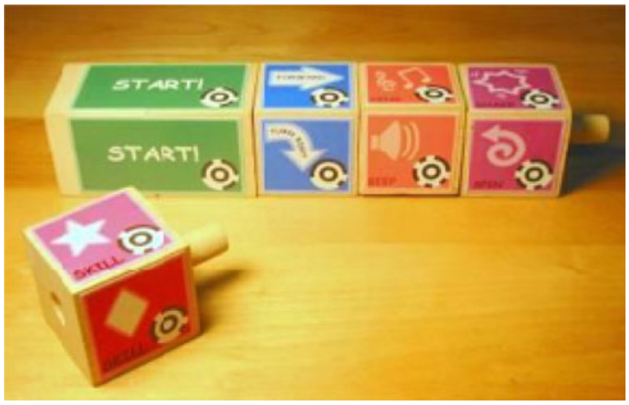

Fig. 1 The Tern tangible programming languages allow children to create computer programs by connecting wooden blocks

tangible and mouse-based computer programming using a between-subject approach, in which participants used only one of the two interaction styles for the duration of the study [3, 10]. The results of the first two studies motivated us to create the first hybrid version of Tern, which combined tangible and graphical interaction into one interface. The third study presented this hybrid prototype to children as part of a weeklong, summer robotics camp. The goal of this third study was to explore how children used the hybrid interface to explore powerful ideas of computation and robotics introduced during the intervention.

We conclude this paper by summarizing the situations in which tangible interaction seems to offer concrete advantages for learning.

2 Tangible programming languages

Since the 1960s a large number of programming languages targeted at novice users have been created [12]. Notable recent languages include Scratch [23], Alice [5], and ROBOLAB [21]. Much of the effort to create these languages has been motivated by the belief that learning to program is not only necessary for *technological fluency* [2], but that it is also beneficial as an academic endeavor in its own right [18]. In other words, that *computational thinking* [30], as a more general abstraction of computer programming, is a powerful cognitive skill that can have a positive impact on other areas of children's intellectual growth. Research has indicated that learning how to program computers can have a positive and measurable effect on children's achievement, not only in areas such as math and science, but also in language skills, creativity, and social-emotional interaction [2, 4].

Kelleher and Pausch [12] offer a taxonomy containing well over fifty novice computer programming systems, a great number of which aim to ease or eliminate the process of learning language syntax, perhaps the most often cited source of novice frustration. Beyond syntax, there are many specific conceptual hurdles faced by novice programmers [4, 13, 22]. A relatively recent approach to ease the learning process has been the creation of *tangible programming languages* [16]. By combining computer programming and tangible interaction, researchers are beginning to explore the exciting potentials of programming in *and* with the physical world. Some ideas that have been generated include the blending of physical space and digital programming [8, 17], robots that are also embodied algorithmic structures [27, 31], the incorporation of found or crafted materials into algorithmic expressions [28], or the integration of physical activity and play with programming [25]. All of this work on tangible programming is pioneering; however, there is notable lack of evidence that tangible systems offer any benefits compared to onscreen counterparts.

The research studies in this paper involve a tangible programming language called Tern [9] (Fig. 1). We designed Tern for use in educational settings to engage children in computer programming and robotics activities. Rather than using a keyboard or mouse to write programs on a computer screen, children instead use a collection of interlocking wooden blocks to build physical computer programs. Each block represents an action, a flow-of-control construct, a parameter, or a sensor value. Tern uses computer vision *fiducials* (the circular, black-and-white symbols printed on each block in Fig. 1) to convert physical programs into digital code [9]. These fiducials allow the computer vision system to identify each block and determine its position in relation to the other blocks. This computer vision system works reliably in a variety of lighting conditions with a standard desktop or laptop computer and a consumer web camera.

3 Hybrid interfaces

In this paper, we consider combining tangible and mouse-based computer programming to create hybrid interfaces. We use the term *hybrid interface* to refer to single interactive system that consists of two or more *equivalent interfaces*. Users have the freedom to select an interface to meet their current needs or preferences, and they may transition from one interface to another at any time, ideally assisted by the interactive system itself. Each interface controls the same digital system, and the various input mechanisms may or may not be synchronized with one another. This differs from augmented- or mixed-reality systems [e.g., 24, 29], which blend the physical and digital worlds but tend to offer a single fixed input mechanism.

We use the term *equivalent interfaces* to refer to interfaces with distinct interaction styles that nonetheless share a common *essential interaction metaphor*. The idea is that designers will select an essential interaction metaphor to assist users in the transition between interaction styles. For example, with Tern, the essential interaction metaphor is one of connecting interlocking blocks—this action represents the process of constructing syntactically correct computer programs. In the tangible system, these programming elements are real wooden blocks, and in the graphical system, they are pictures on a computer screen. In our research, we have found that children as young as 4 years old can grasp this metaphor and easily transition between its graphical and tangible instantiations.

Our use of the term *interaction style* is more general than that proposed in early human–computer interaction literature (e.g., [26]). We use the term to refer to a collection of conventions and techniques that facilitates user interaction. Interaction style is related to but not necessarily equivalent to the specific input and output devices involved in the interface. For example, command line interfaces and graphical user interfaces are two distinct interaction styles that typically involve a mouse, keyboard, and computer screen. Likewise, tangible interaction, as defined by Hornecker and Buur [11], could refer to several distinct interaction styles. For example, an interface that emphasizes the manipulation of physical objects has a different interaction style than one that emphasizes the movement of a person's body in space.

4 Science museum study

Our first study comparing the use of tangible and mouse-based programming languages was conducted at the Museum of Science in Boston (see [10] for a full description of this study and the results). The study involved observations of museum visitors using Tern to program a robot at a museum exhibit.

4.1 Research questions

For this study, we were interested in several research questions related to informal science education:

- Inviting: *Does the type of interface (tangible or graphical) affect how likely visitors are to interact with the exhibit?*
- Active Collaboration: *Does the type of interface affect active visitor collaboration?*
- Apprehendable: *Does the type of interface affect whether or not visitors are able to develop an understanding of how the exhibit works?*
- Engaging: *Does the type of interface affect how long visitors interact with the exhibit?*
- Visitor Computer Programs: *Does the type of interface affect either the number or the complexity or programs that visitors create to control the robot?*

4.2 Methods

For the purposes of the study, we created two experimental conditions, tangible and graphical. In the tangible condition (TUI), visitors create programs using physical wooden blocks shaped like jigsaw puzzle pieces (Fig. 2). In the graphical condition (GUI), visitors create programs on a computer screen using a single computer mouse. We designed graphical and tangible conditions to be as similar as possible, using the essential metaphor of connected jigsaw puzzle pieces. All other aspects of the physical exhibit installation remained the same. During the study, we set up only one of the two interfaces for visitors to use on a given observation day, and we alternated conditions on subsequent days. Visitors used the exhibit without interacting with researchers and without help from the museum staff.

4.2.1 Measurements

To measure the *inviting* quality of the exhibit, we kept a tally of the number of people who noticed (looked or glanced at) the exhibit within a five-foot radius of the installation. Of the people who noticed the exhibit, we recorded the number of people who touched the exhibit with their hands. The time that a visitor first touched the exhibit was recorded as the start of a session. Session data were recorded on a per-group basis.

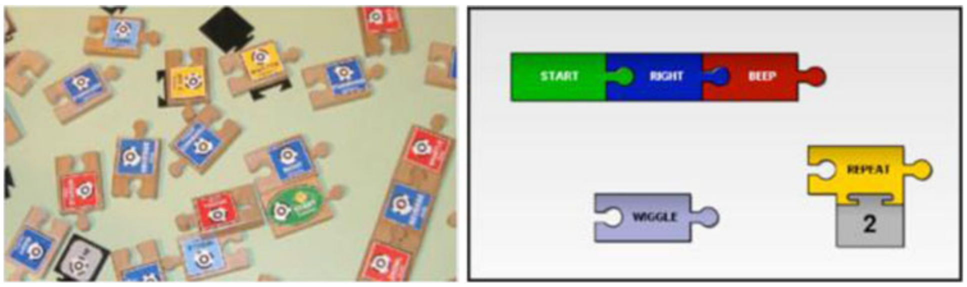

Fig. 2 At the museum exhibit, visitors create programs using wooden blocks shaped like jigsaw puzzle pieces (*left*). The graphical condition (*right*) preserves the essential metaphor of connected jigsaw puzzle pieces

To measure *active collaboration*, we compared the number of active participants to the number of passive participants in each interaction session. An active participant is someone who touches or physically interacts with the exhibit in some way, while a passive participant is a visitor who simply observes other members of his or her social group using the exhibit.

To measure *apprehendability*, we noted whether or not a social group was able to develop an understanding of how the exhibit worked. In other words, did visitors understand that pressing the run button caused the robot to execute a program? For our purposes, programming the robot one time was not sufficient evidence of understanding. Instead, we required evidence that visitors were purposefully putting pieces together to create more than one program. Specifically, they had to press the run button more than once, and they had to rearrange the program at least once.

To measure *engagement*, we recorded the duration of each group interaction session. This was recorded as the time the first group member started interacting with the exhibit to the time that the last group member left the exhibit.

Finally, to analyze *visitor computer programs,* we configured the exhibit computer to log every program compiled by participants. This was in the form of a screen shot for the GUI condition and an image captured by the digital camera for the TUI condition. In analyzing these logs, we were interested in three questions: does the type of interface affect (1) the number of programs created by visitors per session; (2) the length of programs created; and (3) the complexity of programs created?

4.2.2 Participants

We observed a total of 260 individuals at the Museum of Science (108 for the GUI condition and 152 for the TUI condition). Of these, 104 of the participants were children (47 for the GUI condition and 58 for the TUI condition). We defined a child as an individual 16 years old or younger. However, for these observations, we did not interview the visitors, so our participant ages are estimates. Several of our measures involve observations of social groups. In the GUI condition, there were 25 such groups, 16 of which were family groups, containing at least one adult and one child. In the TUI condition, there were 39 total groups, 18 of which were family groups. In analyzing visitor computer programs, we looked at programs created during the first 2 days of observations (1 day for each condition). This included 13 groups in the GUI condition and 20 groups in the TUI condition.

4.3 Results

For two of our research questions (inviting and active collaboration), we found significant differences between the two conditions. In terms of being inviting, we hypothesized that the use of familiar objects (wooden blocks) would transform an unfamiliar and potentially intimidating activity like computer programming into an inviting and playful experience. For the graphical condition, 33 of the 108 visitors (30.6%) who noticed the exhibit stopped to try it. And, for the tangible condition, 78 out of 152 visitors (51.3%) who noticed the exhibit stopped to try it. A two-tailed z-test resulted in a significantly difference between the two proportions, $z = 3.34$, $p = 0.0009$. This difference was especially pronounced for children and for girls in particular. For the graphical system, 33.3% of girls who noticed the exhibit also tried it. This number rose to 85.7% when the tangible system was presented. Figure 3 shows the percentage of who interacted with the exhibit after noticing it by age and gender.

For active collaboration, we expected that the tangible blocks would provide better support for simultaneous active participants. Our results confirmed this expectation. The average percentage of active participants in the 25 groups in the graphical condition was 59.9% (SD = 22.6), while the average in the 39 groups in the tangible condition was 82.8% (SD = 24.7). We conducted an independent t-test and found a significant difference between the means, $t(62) = 3.75$, $p = .0002$. The average group size in the graphical condition was 2.44 (SD = 1.08), while the average groups size in the tangible condition was 2.56 (SD = 1.45).

For our remaining research questions, we found no significant differences between the two conditions. For apprehendability, of the 25 groups that we observed in the graphical condition, 18 (72%) successfully developed an understanding of how the exhibit worked, while in the

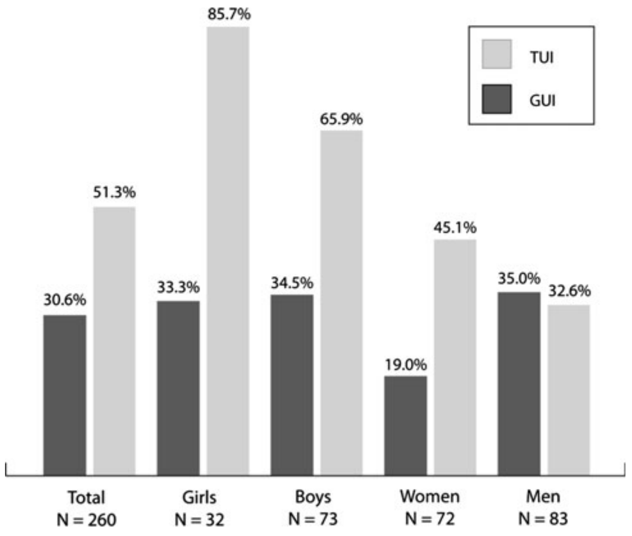

Fig. 3 Percentage of visitors who interacted with the exhibit after noticing it by age and gender

tangible condition, 29 out of the 39 (74%) groups were successful ($z = 0.21$, $p = 0.83$). For engagement, visitors spent an average of 4.02 min interacting with graphical condition (SD = 4.87) and 5.03 min with the tangible condition (SD = 5.84). Results of a *t*-test showed no significant difference between the means, $t(62) = 0.72$, $p = 0.475$. We did, however, observe a significant difference in average session length between groups with only a single active participant (2.2 min, SD = 1.99) and groups with multiple active participants (7.0 min, SD = 6.59), regardless of the interface condition, $t(62) = 3.64$, $p = 0.0006$. This suggests that for engagement, the type of interface might be less important than actively involving multiple participants. Finally, for visitor computer programs, despite our expectations that visitors would create longer and more complex programs with the tangible interface, we found no significant differences between the conditions for any of our measures.

4.4 Summary

This study suggests some ways in which tangible interaction might be advantageous for informal science education. In particular, we found that the use of wooden blocks was more inviting for visitors and encouraged (or facilitated) simultaneous active participation. However, once visitor groups decided to try the exhibit, they spent roughly equal amounts of time interacting, and they produced similar programs in both conditions.

5 Kindergarten classroom study

For the second study, we conducted an eight-week intervention in a suburban elementary school [3]. This study was guided by the following questions: given access to different styles (i.e., TUI vs. GUI) of age-appropriate technology, are young children capable of programming their own robotics projects without direct adult assistance? At the same time, to what extent do young children understand the underlying powerful ideas behind computer programming?

5.1 Methods

We collected qualitative data in the form of observation notes, photographs, and videotape. We also collected student work (both programming code as well as the robotic artifacts) and conducted one-on-one semi-structured interviews with a sample of children and teachers in the classroom. Three researchers were involved in this study: two collected data, while the third acted in the role of lead teacher.

Table 1 Breakdown of classrooms and participants for each condition

Condition	Time of day	# Girls	# Boys
TUI	Morning	11	7
GUI	Morning	11	9
TUI	Afternoon	11	7
GUI	Afternoon	8	10

For this study, we developed an 8-h curriculum unit designed to introduce a series of *powerful ideas* [2, 19] from computer science in a structured, age-appropriate way. The curriculum was made up of whole-class instruction, small-group challenge activities, and open-ended student projects. We provided students with pre-assembled robot cars to teach preliminary computer programming concepts. These cars were built using LEGO Mindstorms RCX construction kits. As the unit progressed, students disassembled these cars to build diverse robotic creations that incorporated arts and craft materials, recycled goods, and LEGO parts. All classes were taught in the school's science activity room. In this room, there were four desktop computers in a back corner of the classroom for students to use. In addition, there was an LCD projector available and a large projection screen in the front of the classroom.

5.1.1 Participants

The study involved four classrooms of kindergarten children, including 74 children (ages 5–6) and eight teachers. In attempt to understand the implications of tangible versus graphical programming in kindergarten, we divided these four classrooms into two groups. Two of the classrooms used sets of the tangible programming blocks (TUI) that we redesigned for younger children. The other two classrooms used an equivalent graphical programming language (GUI). Table 1 summarizes the four classrooms involved in this study.

5.2 Results

For our first research question, we were interested in whether young children could create their own computer programs. Based on observation notes and an analysis of videotape, we found that children were able to manipulate the tangible blocks to form their own programs. For children in the GUI condition, however, we observed a range of capabilities in terms of being able to manipulate the computer mouse. For some children, the process of building graphical programs was tedious, while for other children, the mouse-based interface seemed easy to use.

Beyond physical manipulation, we observed that children were able to understand the icon-based programming syntax, even if they could not read. We found that most students were able to differentiate the blocks and discern their meanings in both the GUI and TUI conditions. While not all of the children could read the text labels on the blocks, we saw evidence that children were able to use the icons as a way to interpret meaning. For example, in the initial sessions, we asked children to look at the blocks and describe what they saw. Some children were simply able to read the text labels on the blocks. Other children said things like: "mine says arrow" or "mine says music." In reality, the text on the blocks reads "Forward" and "Sing." We used whole class games to reinforce the meaning of each block in terms of the robot's actions.

The children also seemed to understand that the blocks were to be connected in a sequence and interpreted from left to right. For example, one student in an introductory TUI session pointed out, "they connect; some have these [pegs] and some have these [holes]." Another student added, "the End block doesn't have one [peg]," with another classmate adding, "the Start block doesn't have a hole." Likewise, in the GUI classes, we saw no evidence that children were confused by the visual interaction metaphor.

For our second research question, we were interested in how children understood the concepts introduced by the curriculum. As the unit progressed, we moved beyond programs consisting of simple sequences of actions and introduced more advanced constructs such as loops, sensors, and numeric parameter values. Through these activities, we found evidence that children could engage these concepts, reasoning about possible outcomes of different blocks to form, and test solutions to challenge activities. For example, in one activity, we prompted teams of four students with this challenge: *Your robot is hungry. Can you program it to drive to the cookies?* A picture taped on the wall approximately four feet from the robot represented the cookies. Each Forward block would cause the robot to drive forward approximately one foot.

After the challenge activity, one of the researchers discussed students' solutions with one of the GUI classrooms. One group determined that they needed eight forward blocks to reach the cookies. After learning the repeat syntax, they realized that largest parameter value for a repeat block was five, so they needed 3 more forward blocks. The students then suggested this program:

$$\text{Repeat}\,(5) \rightarrow \text{Forward} \rightarrow \text{Forward} \rightarrow \text{Forward} \rightarrow \text{End Repeat}$$

When we tested to see if that program would work, the students saw that the robot actually went forward 15 times. One student noticed that 15 was a "5 number," and another said it was the 3rd "5 number."

Later in the unit, we introduced students to the idea of sensors through a *Bird in the cage activity*. Here the cage was a cardboard box and the bird was a robot with a light sensor attached. The lead teacher told a story of a bird who liked to sing when released from her cage. We used this program to have the robot act out the story.

$$\text{Begin} \rightarrow \text{Repeat (forever)} \rightarrow \text{Wait for Light} \rightarrow \text{Sing} \rightarrow \text{End Repeat} \rightarrow \text{End}$$

The students were curious how the robot was able to sing when it was removed from the cage. We used this curiosity to prompt students to explore the idea of sensors and to develop hypotheses about how the robot is able to sing when it emerges from the box. Finally, we demonstrated the program that controls the robot. The following transcript is from one of the TUI condition classrooms:

Teacher: Do you recognize all of the blocks?
Student 1: No. The moon one. [Wait for Dark]
Student 2: The sun and moon [Wait for Light]
Teacher: Can one of you read this?
Student 2: Wait for Light
Student 1: It means you're going to wait for the dark to come.
Teacher: What are these?
Students together: Repeat
Teacher: What do they do?
Student 3: Start all over again.
Teacher: The bird is outside.
Student 2: The witch catches the bird
Student 1: If we turn this block over we could make him sing when it's dark outside. It might be shy so he sings in the dark.

Here the child was pointing out that it would be possible to use a Wait for Dark block instead of a Wait for Light to achieve the opposite effect, a bird that sings only when it is inside the cage—alternating sides of the same cube had Wait for Light and Wait for Dark labels.

For the remainder of the curriculum unit, the children worked in pairs to create robotic creatures that ideally incorporated one moving part and one sensor. The children struggled with many aspects of this activity and required substantial help from teachers. This difficulty might have reflected a limitation of our curriculum, which did not include time for children to explore the RCX brick and understand its role in robotic constructions. As part of the final project presentations, we asked students not only to demonstrate their robots, but also to show the class the blocks they used to program it. In many cases, students selected blocks that had little to do with their actual programs. In other cases, however, students were able to recreate programs more accurately. For example, in this transcription, two girls described their toucan robot:

Teacher: So what does it do?
Student 1: We have to make the program first.
Student 1: [Starts connecting blocks] Begin. Wait for Dark…
Teacher: Can you tell us what we're doing while you're doing it?
Student 2: Yes. I think we should use a repeat block.
Student 1: Great idea
Teacher: E., why don't you show us your robot while S. builds the program.
Student 2: This is our robot… [she starts to demonstrate how it works]
Student 1: The program isn't done yet!
Student 2: [Looking at the blocks] It's supposed to repeat the shake
Student 1: Yes. It's going to repeat the shake over and over again.

[The program student 1 has created is:]

Begin → Wait for Dark → Beep → Turn Left → Repeat → Shake → End

Student 2: [Runs the program on the robot for the class to see. The actual program waits for dark, turns, beeps, and then shakes once.]
Student 1: These are the three blocks we used [She puts these blocks in the middle of the rug: Begin, Wait For Dark, Beep]

Here it is clear that there is some confusion on the part of the students about the concept of a program being stored on the RCX robot. However, the blocks the students chose to explain their robot's behavior are consistent with the blocks they used to program the robot earlier that day. Moreover, they seemed to understand the notion of repeating an action. And, although there was no verbal evidence, the first student seemed to understand the notion of waiting for dark given her correct inclusion of that block in her program.

5.3 Summary

In terms of the first research question, our work with both a tangible and a graphical condition suggests that while many young children do struggle with the mouse-based interfaces, graphical languages can nonetheless serve a useful role in the classroom. For certain activities and for certain children, the tangible version of Tern was clearly advantageous. On the other hand, the most independent student work that we observed was done with the graphical interface. In many cases, we believe that other factors such as teacher involvement and time of day overshadowed the effects of a particular interaction style on the quality of students' activities.

Thus, one conclusion that we drew from this study is that while there were many situations for which it was advantageous to use physical blocks, it is worthwhile to support the integration of tangible and graphical (and perhaps even text-based) programming in learning environments. Providing multiple representations affords educators and learners the flexibility to choose the most appropriate tool for a given situation. It also opens exciting new possibility for programming language design—imagine, for example, being able to create a subroutine in a graphical system that is then embodied in a physical block. One way to think about this motivation is to consider the variety of participant structures [20] commonly used in kindergarten classrooms. For example, in whole-class activities, the teachers would sit with the entire class on a large rug in an open area of the classroom. We used these situations to introduce new programming concepts, to model activities for students, or to consolidate knowledge after an activity. For the graphical condition, we projected the programming environment on a large screen in the front of the room. The graphical condition worked, but it imposed additional constraints on the classroom structure. The children had to be oriented around the fixed projection screen, and often the lights in the classroom needed to be dimmed. In contrast, the tangible interface offered a more flexible alternative. Children could sit in a circle and focus on the tangible blocks that would either be passed out to children or collected in a pile in the middle of the rug.

Other participant structures included individual work, small-group work (2–3 students), and large-group work (4–6 students). There were also situations in which teachers would interact with individuals or groups of students, constituting additional participant structures. Many factors could influence student/teacher preference for one interaction style or another in these settings, including such things as the fine motor skills of individual students and the willingness of group members to share single user input devices offered by the graphical system. Likewise, some students found the graphical system easier and quicker to use, or in some cases, a more sophisticated or "grown up" style of interface.

6 Hybrid tern interface

Based on results from our first two studies, we developed an initial hybrid prototype of Tern in the spring of 2009. This prototype is a single computer program that runs on a desktop or a laptop computer. Users can elect to create programs either using physical wooden blocks or using graphical programming elements from an onscreen tool palette (Fig. 4). When a child compiles a tangible program (by pressing the space bar or by clicking on a button on the

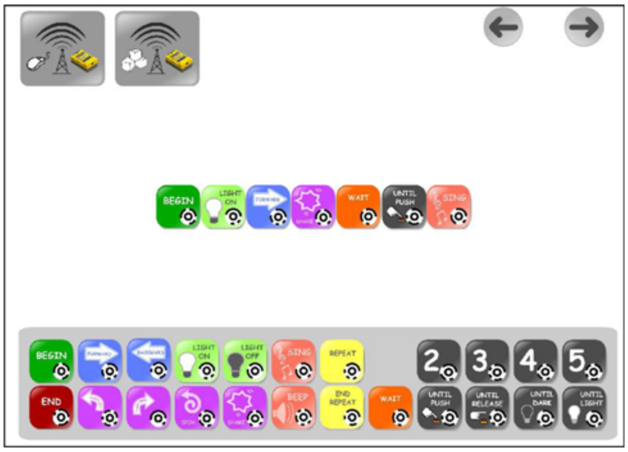

Fig. 4 A screenshot of the Tern hybrid prototype. Users can select to program graphically or tangibly

computer screen), the system takes a photograph of the program and converts it into digital code. The system then displays this photograph onscreen alongside a graphical version of the same program. To revise the program, the child can either rearrange the physical blocks or rearrange the virtual blocks on the screen. In this way, the system provides an automatic translation from a tangible to a graphical representation. To translate from a graphical program to a tangible program, however, the child must manually copy the onscreen program using the wooden blocks.

7 Robotics summer camp study

To evaluate the hybrid Tern implementation and to uncover any usability problems, we conducted an evaluation with 15 children entering kindergarten through second grade during a weeklong summer robotics camp at Tufts University.

7.1 Research questions

Our research questions centered around three major topics:

- Comprehension: *How deeply do children understand the powerful ideas of computation and robotics that are introduced through the intervention?*
- Collaboration: *How often do children interact with one another while working on their robotics projects?*
- Interaction: *How do children engage with each of the interaction styles offered by the hybrid interface (tangible and graphical)?*

7.2 Methods

During the camp, children worked together in freeform groups to build a robotic city using LEGO Mindstorms construction kits. They designed and built robots based on their own interests and programmed them using the hybrid Tern interface. Each day, we introduced a robotic programming concept followed by a small-group challenge activity that encouraged the use of these new concepts. The goal of these small-group activities was to have children practice and solidify their understanding of the new concepts and then apply them to their own projects later in the day.

7.2.1 Measures

To assess the children's *comprehension* of computer programming and robotics concepts, the researchers conducted semi-structured interviews with the children about their projects throughout the workshop. The researchers also administered a 10-question assessment at the beginning and the end of the week. Each child was asked a series of ten "yes" or "no" questions such as: "Robots can think by themselves" and "Robots are alive." The complete list of questions is shown in Table 2.

To measure children's *collaboration* on their robotic projects, we developed an instrument that we call a collaboration web or "thank you" web. At the beginning of each day, the children were presented with a personalized printout with their photograph in the center of the page and the photographs and names of all other children in the class arranged in a circle surrounding their own photo (Fig. 5). Throughout the day, each time a child collaborated with another child (where "collaboration" is defined as getting or giving help with a project, programming together, lending or borrowing materials, or working together on a common project), the child was told to draw a line from his or her own photo to the photo of the collaborator.

To understand children's *interactions* with the two versions of the Tern interface (tangible and graphical), all programming activity on the laptop PCs was logged. At the end of the session, these logs were analyzed to determine how the children interacted with Tern during their free

Table 2 Pre/post-assessment questions

1. Robots are machines
2. All robots are made of the same materials
3. Robots must have moving parts
4. Robots can think by themselves
5. All robots look like alike
6. Robots must be able to move around the room
7. Robots are operated using remote controls
8. People tell robots how to behave using a list of instructions called a program
9. Some robots can tell what is going on around them
10. Robots are alive

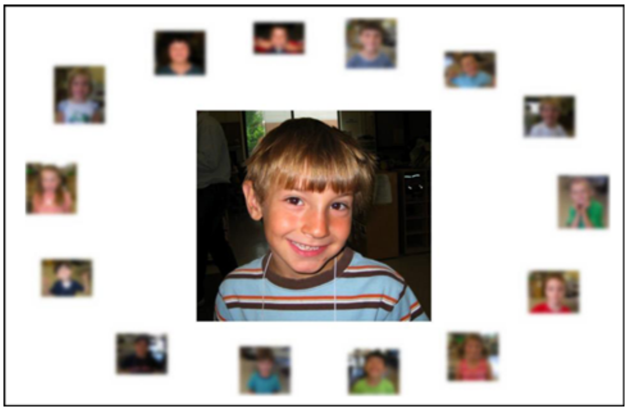

Fig. 5 A "thank you" web for one of the camp participants

building and programming time. In particular, our analysis sought to uncover usage patterns that emerged between the two interaction styles.

7.2.2 *Participants*

A total of 15 participants (ages 5–7) enrolled in the summer camp (12 boys and 3 girls). Seven children were entering kindergarten (6 boys and 1 girl), four were entering first grade (2 boys and 2 girls), and four were entering second grade (all boys).

7.3 Results

Our first research question sought to explore how deeply the children understood the computer programming and robotics concepts that we introduced. We scored the pre/post-assessment into three categories: *Needs to Develop* (0–2 correct responses), *Developing* (3–6 correct responses), and *Developed* (7–10 correct responses). Seven of the children scored in the *Developed* range on both the pre-and post-assessment; four improved from *Developing* on the pre-assessment to *Developed* on the post-assessment; two improved from *Needs to Develop* to *Developing*; and the remaining three children scored in the *Developing* range on both the pre- and post-assessment. By the end of the program, all children were able to correctly identify the following characteristics of robots:

- *Robots are machines.*
- *Robots cannot think by themselves.*
- *Robots are told what to do using a list of commands called a program.*
- *Robots can use sensors to get information from their environment.*
- *Robots are not alive*

In interviews, children revealed that their projects involved the use of many of the ideas introduced throughout the week. For example, one child explained that his robotic drawbridge used a touch sensor to signal that the bridge should go up. He explained, "the bridge waits until a boat comes by" (here he brushed the touch sensor with his finger to simulate a passing boat) "and then the bridge knows it needs to go up." The final projects were designed and implemented by the children with minimal intervention from the camp coordinators. These projects included racecars that used infinite loops to move forward, a button-operated factorywithaconveyorbelt,andahumanoidrobotthatwavedher armswhen "tickled."

Analysis of interviews indicated that the children involved in the program were engaged in active exploration of robotics and had become acquainted with many concepts from this domain. For example, one student noted that the button that makes the double doors in the building accessible to people with disabilities "works like a touch sensor." He noted that because the door does not move until someone presses the button, "it must have a Wait For block" in its program.

For our second research question, results of the collaboration web analysis indicated that each child collaborated with an average of 4.5 other participants each day (with a reported low of 2 and a reported high of 7).

Our third research question sought to understand patterns of interaction with the hybrid interface. One pattern that emerged from the log analysis provided evidence that children prefer to use the graphical interface for what we refer to as "rapid prototyping." In this interaction style, children first compile a program using either the tangible or graphical interface. There is then a one- to three-minute gap in the log where there is no activity, after which a new program is compiled using the graphical interface that is similar to the first program, with the addition or subtraction of at most two blocks or containing some permutation of the original blocks. From this logged interaction, we infer that during the short gap between compiles, the child was observing his or her robot as it executed the program, noting areas that needed slight modifications, and then going back to the computer to make these modifications using the graphical interface. Of the 10 computer logs that were analyzed, 8 contained this behavior. We found it particularly interesting that in all 8 examples of rapid prototyping behavior, children utilized the graphical interface to make their minor modifications regardless of their original programming method.

Another pattern that appeared in the data analysis was the tendency for children to program using the tangible interface earlier in the morning, gradually moving over to the graphical interface later in the session. This aligns with the curriculum schedule of introducing a new concept and its corresponding block and icon at the group meeting upon the children's arrival in the morning. This suggests that the children in this program might have felt more comfortable exploring a new programming concept using the tangible blocks and shifted toward the graphical interface as they became more secure in their understanding.

Some children elected to utilize only one of the interaction styles for the duration of the workshop. For example, one child whose guardians indicated that he found laptop trackpads particularly engaging elected to exclusively use the graphical interface. Two other children, who were among the youngest in the program elected to use the tangible blocks for the majority of their programming because they found the mouse difficult to maneuver in a reliable manner.

8 Discussion

Our emphasis in this paper is to propose that it is advantageous to combine tangible interaction with more traditional interfaces to create hybrid systems. Based on our research, we believe that this leads to several immediate advantages. The most important of these is that it gives actual participants in learning environments—teachers, students, museum staff, etc.—the freedom to select the type of interaction style most appropriate for a given learning situation. This flexibility is especially important in classroom settings, where teachers are often determine if and when a particular technology will be used [6]. In addition, the use of hybrid interfaces means that not every feature of a software system needs to be implemented tangibly. For example, saving student work on a file system and setting precise parameter values are features that might be better left to the graphical version of the interface. In a similar vein, it is much easier to deploy a pure software system (e.g., over the Internet) than to deploy a tangible system. Thus, a teacher might be able to download and try out the graphical side of a hybrid interface before investing in the tangible components.

Beyond these immediate advantages of hybrid interfaces is the potential to provide layered scaffolding to students as they progress toward increasingly authentic programming environments. For example, students might start with a tangible system and then transition to a graphical system with increased capabilities and complexity. Taking this idea further, students might later transition from a graphical programming environment to a text-based environment with advanced features and capabilities. In this scenario, students would progress through a *tiered hybrid interface*. For a tiered hybrid interface, the essential interaction metaphor connecting the tangible and graphical interfaces could be different from the essential metaphor connecting the graphical and text-based interfaces. Other interaction techniques might fit this model as well. For example, many multi-touch tabletop devices can recognize tangible input by means of computer vision fiducials. Such interaction platforms might provide an even more direct bridge between tangible and graphical interaction.

8.1 Advantages of tangible interaction

In exploring tangible, graphical, and hybrid systems, we have identified several potential advantages of tangible interaction that might merit further investigation. In our museum study, we found that our tangible interface was more inviting and better at supporting active collaboration. This result was especially important because the tangible interface seemed to appeal to girls as well as boys, which was not the case with the graphical interface in the museum. An open question is whether these effects translate into learning environments where students have some choice in the activities in which they participate (for instance, a hybrid interface used in a classroom).

In classroom environments, the tangible interfaces seemed to be advantageous for whole-class activities—students can sit in a circle and refer to a shared set of physical objects. These activities can be conducted in different locations without the need for students to gather around a large shared display or projection screen. For young children with developing fine motor skills, the tangible blocks have an obvious advantage in that they allow participation in an activity that might otherwise be impossible. We should point out that Tern does not take full advantage of interaction with physical objects in the world. For example, rich actions like shaking, turning, squeezing are meaningless to our system. Thus, there are many potential pedagogical advantages that we have not considered.

9 Conclusion

In this paper, we have reviewed three research studies involving the use of programming languages with children in a variety of learning settings. The first two studies—conducted in a science museum and kindergarten classrooms—took a between-subject approach, in which children used either a tangible or a graphical interface to create computer programs. Through these studies, we identified situations in which tangible interaction seems to offer advantages for learning. For example, in classrooms, the tangible interface seemed especially well suited for whole-class activities and discussions. Likewise, in museums, the tangible blocks seemed to do a better job of enticing children to explore programming activities. In other situations, however, the graphical interface seemed more appropriate. For example, some of the most productive small-group work that we observed in kindergarten classrooms involved the graphical interface. Given these findings, we decided to explore the idea of creating hybrid tangible/graphical interfaces that give learners and educators the freedom to select and interaction style to meet their needs or preferences as dictated by the situation. After building an initial hybrid prototype, we evaluated it with children in the setting of a weeklong robotics camp. Findings from this final evaluation suggest ways in which children interact with hybrid tangible interfaces. We observed that children were able to fluidly transition from one interaction style to the other and that they seemed to prefer the tangible interface for early exploration and the graphical interface for subsequent "rapid prototyping."

Acknowledgments We thank Robert Jacob and our collaborators at the Tufts University Human–Computer Interaction Laboratory. Work at the Boston Museum of Science was the result of a collaboration with Taleen Agulian, Dan Noren, Lucy Kirshner, Erin Solovey, Bill Rogers, and Tim Moore. We also thank Rachael Fein and Emily Lin for their invaluable assistance in our kindergarten classroom research. We thank all students in the DevTech research group, especially Elizabeth Kazakoff, Louise Flannery, and Ken Lee. Finally, we thank the National Science Foundation for its support of this research through grants IIS-0414389 and DRL-0735657. Any opinions, findings, or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Antle A, Droumeva M, Ha D (2009) Hands on what? Comparing children's nouse-based and tangible-based interaction. In: Proceedings interaction design and children 2009, Como, pp 80–88
2. Bers MU (2008) Blocks to robots: learning with technology in the early childhood classroom. Teachers College Press, New York
3. Bers MU, Horn MS (2009) Tangible programming in early childhood: revisiting developmental assumptions through new technologies. In: Berson IR, Berson MJ (eds) High-tech tots: childhood in a digital world. Information Age Publishing, Greenwich
4. Clements D (1999) The future of educational computing research: the case of computer programming. Inf Technol Child Educ 1:147–179
5. Conway, M, Pausch, R, Gossweiler, R, Burnette, T (1994) Alice: a rapid prototyping system for building virtual environments. In: Proceedings CHI 1994. ACM Press, Boston, pp 295–296
6. Cuban L (2001) Oversold and underused: computers in the classroom. Harvard University Press, Boston
7. Fails, J, Druin, A, Guha, M, Chipman, G, Simms, S, Churaman, W (2005) Child's play: a comparison of desktop and physical interactive environments. In: Proceedings: interaction design and children 2005, Boulder, pp 48–55
8. Fernaeus, Y, Tholander, J (2006) Finding design qualities in a tangible programming space. In: Proceedings CHI 2006. ACM Press, Montreal, pp 447–456
9. Horn, M, Jacob, RJK (2007) Designing tangible programming languages for classroom use. In: Proceedings tangible and embedded interaction 2007. ACM Press, Baton Rouge, pp 159–162
10. Horn, MS, Solovey, ET, Crouser, RJ, Jacob, RJK (2009) Comparing the use of tangible and graphical programming interfaces for informal science education. In: Proceedings CHI 2009. ACM Press, Boston, pp 975–984
11. Hornecker, E, Buur, J (2006) Getting a grip on tangible interaction: a framework on physical space and social interaction. In: Proceedings: CHI 2006. ACM Press, Montreal, pp 437–446
12. Kelleher C, Pausch R (2005) Lowering the barriers to programming: a taxonomy of programming environments and languages for novice programmers. ACM Comput Surv 37(2):83–137
13. Levy S, Mioduser D (2008) Does it "want" or "was it programmed to…"? Kindergarten children's explanation of an autonomous robot's adaptive functioning. Int J Technol Des Educ 18:337–359
14. Marshall P (2007) Do tangible interfaces enhance learning? In: Proceedings tangible and embedded interaction 2007, Baton Rouge, pp 163–170
15. Marshall P, Price S, Rogers Y (2003) Conceptualising tangibles to support learning. In: Proceeding interaction design and children 2003, Preston, pp 101–109
16. McNerney T (2004) From turtles to tangible programming bricks: explorations in physical language design. J Pers Ubiquitous Comput 8:326–337
17. Montemayor J, Druin A, Farber A, Simms S, Churaman W, D'Amour A (2002) Physical programming: designing tools for children to create physical interactive environments. In: Proceedings CHI 2002. ACM Press, Minnesota, pp 299–306
18. Papert S (1980) Mindstorms: children, computers, and powerful ideas. Basic Books, New York
19. Papert S (2000) What's the big idea? Toward a pedagogy of idea power. IBM Syst J 39(3–4):720–729
20. Philips SU (1972) Participant structures and communicative competence: warm springs children in community and classroom. In: Cazden CB, John VP, Hymes DH (eds) Functions of language in the classroom. Teachers College Press, New York
21. Portsmore M, Rogers C, Cyr M (2000) Integrating the Internet, LabVIEW, and LEGO bricks into modular data acquisition and analysis software for K-College. In: Proceedings American Society for engineering education annual conference & exposition, St. Louis
22. Rader C, Brand C, Lewis C (1999) Degrees of comprehension: children's understanding of a visual programming environment. In: Proceedings CHI 1999. ACM Press, Atlanta, pp 351–358
23. Resnick M (2007) Sowing the seeds for a more creative society. Learn Lead Technol 35(4):18–22
24. Rogers Y, Scaife M, Gabrielli S, Smith H, Harris E (2002) A conceptual framework for mixed reality environments: designing novel learning activities for young children. Presence 11(6):677–686
25. Scharf F, Winkler T, Herczeg M (2008) Tangicons: algorithmic reasoning in a collaborative game for children in kindergarten and first class. In: Proceedings interaction design and children 2008, Evanston, pp 242–249
26. Schneiderman B (1992) Designing the user interface, 2nd edn. AddisonWesley, Reading
27. Schweikardt E, Gross M (2008) The robot is the program: interacting with roblocks. In: Proceedings tangible and embedded interaction 2008, Bonn, pp 167–168
28. Smith A (2008) Handcrafted physical syntax elements for illiterate children: initial concepts. In: Proceedings interaction design and children 2008, Evanston
29. Underkoffler J, Ishii H (1999) Urp: a luminous-tangible workbench for urban planning and design. In: Proceedings CHI 1999. ACM Press, Pittsburgh, pp 386–393
30. Wing J (2006) Computational thinking. Commun ACM 24(6): 33–35
31. Wyeth P (2008) How young children learn to program with sensor, action, and logic blocks. J Learn Sci 17(4):517–550