Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 598547, 10 pages
http://dx.doi.org/10.1155/2013/598547

Research Article

E-Block: A Tangible Programming Tool with Graphical Blocks

Danli Wang,[1] Yang Zhang,[1] and Shengyong Chen[2]

[1] *Institute of Software, Chinese Academy of Sciences, Beijing 100190, China*
[2] *College of Computer Science, Zhejiang University of Technology, Hangzhou 310023, China*

Correspondence should be addressed to Danli Wang; danli@iscas.ac.cn

Received 27 November 2012; Accepted 5 January 2013

Academic Editor: Fei Kang

This paper designs a tangible programming tool, E-Block, for children aged 5 to 9 to experience the preliminary understanding of programming by building blocks. With embedded artificial intelligence, the tool defines the programming blocks with the sensors as the input and enables children to write programs to complete the tasks in the computer. The symbol on the programming block's surface is used to help children understanding the function of each block. The sequence information is transferred to computer by microcomputers and then translated into semantic information. The system applies wireless and infrared technologies and provides user with feedbacks on both screen and programming blocks. Preliminary user studies using observation and user interview methods are shown for E-Block's prototype. The test results prove that E-Block is attractive to children and easy to learn and use. The project also highlights potential advantages of using single chip microcomputer (SCM) technology to develop tangible programming tools for children.

1. Introduction

Papert and Resnick et al. mentioned in their contributions that learning how to program may result in changes to the ways people think [1, 2]. Early studies with Logo also showed that when introduced in a structured way, computer programming can help children improve visual memories and basic numbers senses as well as develop problem-solving techniques and language skills [3]. However, most of the existing programming languages are designed for professionals and are based on texts and symbols which are difficult for children to understand [4, 5]. Hence, previous research works are conducted aiming at lowering the barrier of programming for children [6].

Graphical programming has some significant advantages over textual programming especially in providing visual cues for young programmers [7, 8]. However, the usage of keyboard and mouse which are major input methods in GUI may be difficult for children [9, 10]. What is more is GUI's nature drawback that it falls short of embracing the richness of human's interaction with physical world [11]. This kind of activities contribute to children's learning [12].

Artificial intelligence with tangible programming is kind of feasible programming method for children [13]. Tangible interaction can stimulate multiple senses of children and develop their cognitive abilities [14]. In addition, compared with directly operating computers, using physical objects to interact with computer is much easier to involve children in the process [15]. Instead of using lines of dull codes, the program becomes a collection of physical objects. Children can write programs by assembling the physical objects without keystrokes [16]. Thus, the programming could be more intuitive to children.

Based on our previous work—T-maze [17], this paper proposes a tangible programming tool: E-Block, which is designed for children aged 5 to 9. It defines the programming blocks and the sensors as the input and enables children to write programs to complete the tasks in the computer as Figure 1 shows. The symbol on the programming block's surface is used to help children understand the function of each block. The sequence information is transferred to computer by microcomputers and then translated into semantic information. In E-Block, children need first to find a path for the character to escape the maze in programming stage and

Figure 1: Children with E-Block.

then run the program and trigger sensors when necessary in running stage. In programming stage, when children add a new block into sequence, feedbacks on the screen and block itself will show whether it is placed correctly. Only when it's right, the children can continue programming. After finishing the programming, children could run the program by pressing the start button and enter the running stage. In running stage the character will stop if hits the sensor cells. Children should trigger relative sensors to keep the character going.

2. Related Work

One of the earliest tangible programming projects is AlgoBlock [18]. It adopts several blocks as the interaction medium. Every block has special semantics. Children could write their own program to play a marine game by connecting the objects. RoboTable2 [19] connects tangible programming and graphical programming together on an interactive table which combines several technologies such as camera, projector, computer vision, and blue tooth. It supports the novice to manipulate robots by gestures [17], TUI, and GUI. The whole system is based on the event model by which user needs to define events to trigger certain behavior on the robot. Though the concept of programming in the above works is easy to understand, there still exist some improvements that could be done to the manipulations. The wires between blocks might limit children's activities. Moreover, some complicated scenarios are not suitable for young children.

Tangible programming bricks [20] proposed by MIT with PIC microprocessor built in each physical programming brick communicate with the computer by using a card slot. Children put the bricks into the card slot in different sequences to control the game objects like toy trains and household appliances. This work is highly functional, containing alot of different programming concepts. TurTan [21] is a desktop tangible programming system designed for Turtle Geometry, which is based on Logo. It adds multitouch and gesture recognition into the system. It uses a camera to capture and recognize the real-time position of fingers and objects on the desktop. Then the projector outputs the real-time feedback of the system. Users can control the entire drawing process by manipulating predefined commands in a tangible user interface on the table. The above works require scientific knowledge or contain other concept in the defined commands which might be hard for young children to understand.

Electronic Block [22, 23] uses building blocks to program. It consists of three types of building blocks: sensor block as input, logic block to conduct logic computation, and behavior block as output. It is designed for the preschoolers so that the syntax is simple, easy to manipulate, and free from spatial limitation [24, 25]. Schweikardt and Gross [26] proposes a tangible programming language named roBlock. A processor is installed in each block. The blocks are divided into 4 categories according to their functions: sensor, actuator, logic, and utility. Users can build their own robots by connecting different blocks. Its working principle is quite similar to Electronic Block, so they are all highly functional. Tern [27, 28] is another tangible programming language given by Horn and Jacob and so forth. The programming blocks are made of wooden blocks and every block has specific semantics. Children assemble the blocks to express certain meanings. After writing program with the blocks, children need to manually use camera to capture the block sequence's image that is transferred to computer to identify the information and control virtual roles or real walking robot. Though, the above programming tools are easy to manipulate; however, they fail to support real-time debug and thus offer little help on the programming debug. Once errors are detected, the systems depend on children's own understanding about the task to correct them, which makes programming difficult for beginners.

Based on these previous contributions, we want to develop a system which has the following characteristics. When a child is programming, the system is highly synchronous to the child's latest manipulation. Children are able to place the program blocks with no space limitation. Different kinds of feedbacks will appear on both the computer screen and the tangible programming tools, making benefits for children to position and analyze accurately.

3. Implementation

E-Block was proposed to solve problems in our previous system T-Maze which is based on computer vision technology [29]. In our user study, children were asked to use two systems in order to find the potential advantages of using single chip microcomputer (SCM) technology to develop tangible programming tools of children. Therefore it is needed to briefly describe how T-Maze functions.

3.1. T-Maze System. T-Maze is composed of maze game, programming wooden blocks, camera, and sensor input devices. The maze escaping game, which is the same with the one of E-Block, requires children to control the virtual character in maze to go through relative sensor cells and finally reach the exit of the maze. Children manipulate different wooden blocks to write their own program, which can control the character's moving in the maze. The tool uses a camera to catch the image of the wooden blocks which can be used to

analyze the semantics of children's program. Several problems were found in the T-Maze. First, children are required to program within an area of 25 cm ∗ 30 cm. The programs out of this identification area of cameras could not be captured, which results in children's confusion. Second, the system is based on computer vision which has its inborn drawbacks—camera occlusion. During the test of T-Maze, children could not help raising their hands , blocking between cameras and programs. This caused the delay problem of feedback. Third, feedback is totally shown on screen. Children have to switch attentions between screen and blocks in order to debug. Figure 2 shows T-Maze system.

3.2. E-Block System. E-Block is composed of four parts: the maze game, the programming blocks, the wireless box, and the sensors (see Figure 3). We will introduce each part next.

3.2.1. Maze Game. The virtual maze is composed of four kinds of cells (see Figure 4): start cell, end cell, normal cell, and sensor cell. There is a face on the top left corner of screen to show the real-time feedback. In the programming stage, the character starts from the start cell and children can place the direction block and the sensor block to indicate a path from start cell to end cell. In the running process, the character starts to walk and children have to trigger the relative sensor when the character is stopped by the sensor cell. The rules of this game are listed as follows.

R1: If input is Direction Top Right and the upper right cell of the current location of character is feasible, then give smiley face, green arrow on the screen and blue LED on block.

R2: If input is Direction Top Right and the upper right cell of the current location of character is not feasible, then give sad face on screen and red LED on blocks.

R3: If input is Direction Top Left and the upper left cell of the current location of character is feasible, then give smiley face, green arrow on the screen and blue LED on blocks.

R4: If input is Direction Top Left and the upper left cell of the current location of character is not feasible, then give sad face on screen and red LED on blocks.

R5: If input is Direction Bottom Right and the lower right cell of the current location of character is feasible, then give smiley face, green arrow on the screen and blue LED on blocks.

R6: If input is Direction Bottom Right and the lower right cell of the current location of character is not feasible, then give sad face on screen and red LED on blocks.

R7: If input is Direction Bottom Left and the lower left cell of the current location of character is feasible, then give smiley face, green arrow on the screen and blue LED on blocks.

R8: If input is Direction Bottom Left and the lower left cell of the current location of character is not feasible, then give sad face on screen and red LED on blocks.

Figure 2: T-Maze system.

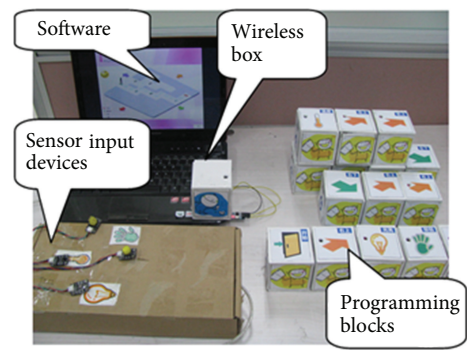

Figure 3: E-Block tool.

R9: If input is Tangible Button and there exist Tangible Button cell nearby, then give smiley face, green arrow on the screen and blue LED on blocks.

R10: If input is Tangible Button and there is no Tangible Button cell nearby, then give sad face on screen and red LED on blocks.

R11: If input is Light Sensor and there exist Light Sensor cell nearby, then give smiley face, green arrow on the screen and blue LED on blocks.

R12: If input is Light Sensor and there is no Light Sensor cell nearby, then give sad face on screen and red LED on blocks.

R13: If input is Temperature Sensor and there exist Temperature Sensor cell nearby, then give smiley face, green arrow on the screen and blue LED on blocks.

R14: If input is Temperature Sensor and there is no Temperature Sensor cell nearby, then give sad face on screen and red LED on blocks.

3.2.2. Programming Blocks. Programming blocks send computer their physical information which is then translated into the program semantics. In E-Blocks, there are four kinds of

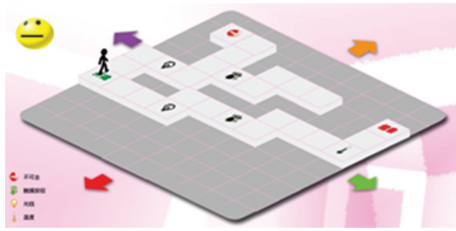

Figure 4: Maze game.

programming blocks: start block, end block, direction block, and sensor block (see Figure 5). As Figure 6 shows, each block has a single chip microcomputer, an infrared transmitter and receiver module, a cell battery module, a wireless module, and an LED. When a new block is added to the array, the former block's infrared signal will activate the new added one. The new added block will send its identity code to wireless box which is connected to PC through USB port. At the same time, it will open its own infrared transmitter. Wireless box will get the physical information from the content and the order of the identity codes received and will then send this information to PC.

(1) Infrared Transmitter and Receiver Module. The adjacent programming blocks transmit data through infrared transmitter and receiver module (VS0038). The former block's infrared signal activates the latter one. After activated, a block will send its identity code to computer by wireless module; at the same time, it will open its infrared transmitter and wait to activate next added block. Computer will get the sequence information from the identity code received. We use Pulse Width Modulation (PWM) to encode the identity code. Similar technology is used in the television remote control.

(2) Wireless Module. The computer communicates with the programming blocks through wireless module. After activated, a block will send identity code to computer by wireless module. Computer will get the sequence of blocks by adding the new identity code to the end of sequence. In the same way, computer will remove the identity code from sequence if the block is removed. If the new added block is not correct, the wireless module will receive a signal from computer and notice the red LED to flash.

(3) LED Module. In the maze map, the gray cells are not reachable. If the child manipulates the character to the unreachable cell, or if the child places a direction block in the sensor cell instead of a sensor block, it will generate an error. If the new added block fits the cell, the blue LED of the block will start to light to distinguish it from the unattached blocks. If the computer finds errors in the programming sequence, the smile face on the screen will turn sad. After that, the computer will find the location of the problematic block and then send a signal to the wireless module of the problematic programming block. As a result, the red LED of the block starts to light to indicate potential error. In this case, children can find the problematic block easily and then try to solve the problem. LED module is on the upper surface of the block so that users can notice it easily as soon as it starts to flash.

(4) Wireless Box. Wireless box is connected to PC through USB port. It will send a request to the programming blocks in turn and wait for their responses. If the wireless box does not receive anything, it means that the programming block is not in the sequence. On the contrary, if the block is added to the sequence, it will send to wireless box its own code. The wireless box will then send the information to PC via USB port. When PC finds some errors in the programming block sequence, it will send a signal to the relative block which will cause the LED to flash.

3.2.3. Sensor Input Devices. Sensor input devices have two functions. Firstly, when children finish programming, they need to press the start button on the sensor input devices to change the programming stage into the running stage. Secondly, in running stage, when the character meets the sensor cell, children need to trigger the relative sensor to keep it going. There are three kinds of sensors as Figure 7 shows: Temperature Sensor, Light Sensor, and Tangible Button Sensor. Each sensor has its own way of being triggered: Temperature Sensor needs to be warmed, Light Sensor shaded, and Tangible Button Sensor pressed. We provide sensor input part because it corresponds to the input part of the program's operation, and at the same time, increases the interests of children and adds more elements to the system.

3.2.4. System Advantages

(1) Fast-Real Time Presentation on PC. Fast real time presentation indicates that it needs about 0.1-0.2 s to change one or more program blocks (produce one or a group of new orders) to PC recognition and finally to the disposal of this group of orders. With the questionnaire, we found that when finishing the placement of a new program block, children would pay attention to the change in the computer screen. In the previous T-Maze experiment, a large number of children responded that the computer processing speed on the program block change was too slow, which resulted in their confusions about the feedback of their manipulations. E-Block solved the problem by the adoption of infrared and wireless technologies.

(2) Fixable and Free Placed Position of Program Block. The effective identification area of E-Block program block is very large. In a range of 4-5 meters, it can effectively and rapidly recognize the program block. When placing the program block, it does not need to be operated in the fixed range of the camera. Children can randomly place blocks in their own favorite positions. At the same time, as long as they are not changing the order, they could move the well-placed blocks to anywhere.

(3) Feedback Appeared in Both Blocks and PC Screen. This function contributes to children's debug of program and leads the children to focus on the task and operation, rather than staring at the smiling face in the upper-left corner of the

(a) Start and end blocks (b) Sensor blocks (c) Direction blocks

Figure 5: Tangible Blocks.

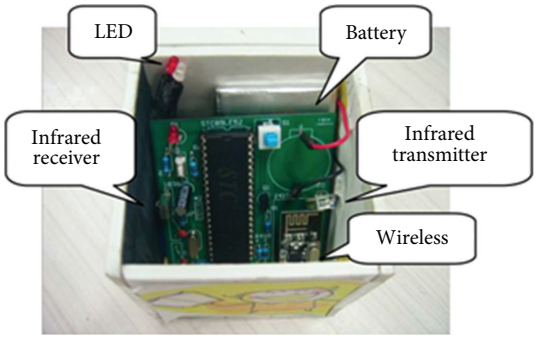

Figure 6: Single chip microcomputer in E-Block.

Figure 7: Sensor input devices.

screen. This feedback mechanism can also act as a trial to the manipulation of information mutualization. Children can place the manipulation to programming, and at the same time the operation result of the program will be shown in the tangibles.

(4) The Role of Sensor. The sensor has the following several effects: correspond to the input part of the operation section in the actual programming, while increasing the interest of children, adding more abundant element for the system. In our tradition programming method, there will be some statements (such as "printf" function in C language) to receive some orders from the keyboard or mouse in the program operation. After the program is operated, we can input the orders or data with these external devices to keep the program operating. So is the role of sensor. At the same time, the sensors can make the system more attractive for children. In the test process, many children showed great interest in the Pressure Sensor and Temperature Sensor devices. The sensors raised children's degree of participation in the running stage of their programs.

4. Use Case

In E-Block, children need first to find a path for the character to escape the maze in programming stage and then run the program and trigger sensors when necessary in running stage. In this part, a simple use case will be described.

4.1. Programming Stage. After successfully choosing a mission, children will start the programming stage. Children must place the start block as the start of the block sequence. Then they need to put the proper direction block into the block sequence which indicates the direction user wants the character to go. When programming the path of the character, children need to place the right sensor block to go on the path when the path hits any sensor cells. If children place wrong programming block in the sequence the smile face on the screen will turn sad. In the meanwhile, the LED in the problematic block turns red so that user can easily find which block is wrong (see Figure 8).

4.2. Running Stage. When children finish the task in programming stage, they need to press the start button on the sensor input devices to turn the E-Block into running stage. And the character will start walking according to the path programmed before. When the character encounters a sensor cell, it will stop until children trigger the relative sensor (see Figure 9).

5. User Study

The user study is a comparison test between E-Block with the former work, T-Maze, which uses computer vision technology to realize the recognition of blocks. It talks about the advantages and disadvantages of SCM-version tangible programming tool, E-Block, compared with the camera-version tool T-Maze.

5.1. Procedure. We ran the user study with a total of 11 children (3 boys and 8 girls) aged 5 to 9. They were asked

Figure 8: A simple case of usage.

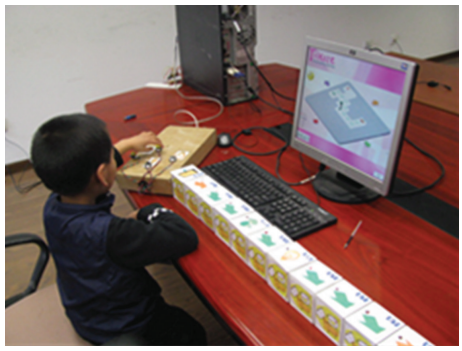

Figure 9: Trigger the relative sensor.

to complete three levels of tasks that need 7, 11, and 12 programming blocks, respectively. The maze-escaping tasks in T-Maze and E-Block were basically the same. The only difference was the blocks they used in the test; T-Maze used computer vision technology and the block was smaller (about 3 cm ∗ 3 cm ∗ 3 cm) with no feedback providing function itself, while blocks of E-Block had LED as feedback and were bigger (about 7 cm ∗ 7 cm ∗ 7 cm). One researcher was available to support children. The study included three stages: explanation and demonstration stage, test stage, and interview stage.

The first stage: explanation and demonstration. We first introduced the composition of T-Maze and E-Block and how to use them. After that, a detailed demonstration video was played together with a real demonstration.

The second stage: test. Children were randomly divided into two groups: 5 children played T-Maze first and then played E-Block. The rest played in the adverse way. We videotaped the whole test process.

The third stage: interview. Once the children completed both T-Maze and E-Block tasks, they were asked to finish a questionnaire and then interviewed individually to gauge their perception on their play experience.

5.2. Coding. Based on the information that we got from interview and tapes, we focused our research on the following aspects: first, whether the tool is easy for children in learning and manipulation; second, what merits are brought by the SCM-version tangible programming tool, E-Block, compared with the camera-version tool, T-Maze, and how these merits influence children's learning and manipulations. For the

Table 1: Coding scheme for behaviors.

Behavior type	Example
Space-related behavior	Move the out-of-boundary programming blocks back to the recognition region
Feedback-related behavior	Look at screen/LED immediately after an addition of a block
Occlusion-related behavior	Put hand between camera and blocks
Programming operation	Add/remove a block to/from sequence of programming blocks

Table 2: Coding scheme for utterances.

Utterance type	Example
Space-related talk	You should keep unused blocks out of the recognition region
Feedback-related talk	Look, the smiling face turns sad; do you notice, the LED is flashing?
Occlusion-related talk	Remember, you should add a block like this; don't put your hand over blocks
Other talk	That's so cool!; I need a green block

11 children being videotaped, we transfer their manipulations together with researcher's guidance manipulations into behaviors and transfer their conversations into utterances.

Because the running state of T-Maze and E-Block is exactly the same, we only use codes to analyze programming state in order to find the differences between them. A total of 300 minutes (132 minutes of T-Maze and 168 minutes of E-Block) were reviewed and annotated by two researchers. Codes were used for analysis if they occurred or not within every five-second interval. Videos were coded into one of four categories: *space-related behavior*, *occlusion-related behavior*, *feedback-related behavior,* or *programming operation,* the numbers of which are shown in Figure 10. Accordingly, we also coded all the utterances consisting of continuous talk without long pauses into one of four categories: *space-related talk*, *occlusion-related talk*, *feedback-related talk, or other talk,* the numbers of which are shown in Figure 11. The full set of codes is shown in Tables 1 and 2. In total, we coded 1106 behaviors and 151 utterances in T-Maze: an average of 33.5 behaviors (SD = 11.24) and 4.58 utterances (SD = 2.08) per task. In E-Block, the numbers are 773 behaviors and 98 utterances in total. 23.4 behaviors (SD = 4.36) and 2.97 utterances (SD = 1.08) per task. We obtained almost perfect

agreements for data of behaviors and 90% agreements for data of utterances based on 100% of the data (Kappa = 1 and 0.87 resp.). We defined space-related and occlusion-related behaviors and utterances to have negative influence on programming because they were to solve the problem caused by hardware and did not contribute to program. We hope E-Blcok can better focus children on programming by reducing such behaviors and utterances. The questionnaire in interview section was a Likert-type scale composed of four questions with punctuation from one to five, one being the minimum and five the maximum score. We analyze the results of video and interview in the following section.

5.3. Results. In this section, we first prove E-Block to be easy for children in learning and manipulation and then start to compare E-Block with T-Maze. The major difference between E-Block and T-Maze is that E-Block enables children to place the block without considering the camera's view scope and occlusion problem. Spatial constraints, occlusion problem, feedback, and programming operation are used as comparison points. They are represented by behaviors and utterances, respectively. We discuss the totals, averages, and proportions for all behaviors and utterances in the programming stage, which leads to a preliminary comparison between tangible programming tools using computer vision technology and single chip microcomputer. We hope this comparison may provide references for future design.

5.3.1. General Results. According to the observation and the questionnaires we collected in the test, we find that every child in our test showed great interest in E-Block. Many children were interested in the sensor and asked us "what is this? How to use it?" Once they saw the character escaping the maze, some said: "Well done!" In the questionnaire, when asked how much they like the game (like very much, like, normal, dislike, dislike very much), 6 children said they liked it very much, 3 selected "like" and 2 selected "normal". When being asked which part they liked most, some children said that they liked triggering the sensor and some said that they are fond of placing the block and manipulating the character to escape the maze. According to the observation, we found that all the children could master the tangible programming block tool quickly. Although some younger children could not place all the blocks in the right order for the first time, we find they were able to adjust the blocks according to the real-time feedback on the screen or the feedback on the block.

5.3.2. Spatial Constraints. In E-Block, children would not need to worry about whether the block is within the range of the camera because they can place them anywhere as long as wireless box can receive wireless signal from wireless module in each block. This distance is about 15 meters from a limit test of E-Block while the programming space of T-Maze is only 25 cm $*$ 30 cm. As defined in Table 1, the average number of space-related behaviors is 3.33 (SD = 1.05) per task in T-Maze and 2 (SD = 0.94) in E-Block. As defined in Table 2 the average number of space-related utterances is 1.67 (SD = 0.58) per task in T-Maze and 0.27 (SD = 0.46) in

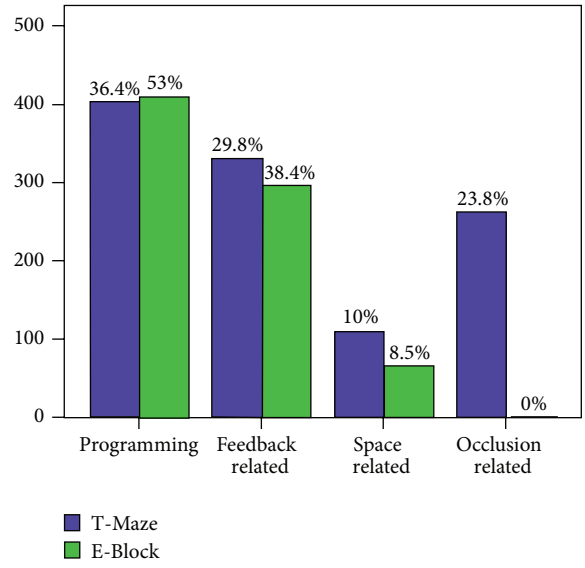

Figure 10: Number of behaviors by category type.

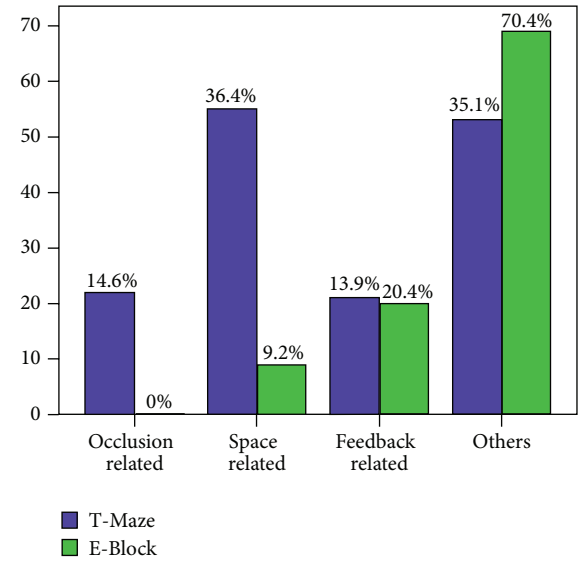

Figure 11: Number of utterances by category type.

E-Block. Figures 10 and 11 demonstrate that such behaviors and utterances reduced significantly in E-Block; however, large size of the programming block also caused some space-related behaviors and utterances. Children had to push the existing block sequence to keep it from going out of desktop. Therefore, we deduce that if the programming blocks in E-Block had had the same size of those in T-Maze, space-related behaviors and utterances would have appeared even fewer. The recognition area of camera-version programming tool is influenced by the performance of camera to a great extent; however, the wireless technology of SCM enables children to manipulate in much larger area.

5.3.3. Occlusion Problem. In E-Block, children had no needs to worry about whether they may block the camera. The

average number of occlusion-related behavior is 7.97 (SD = 2.33) per task in T-Maze and 0 (SD = 0) in E-Block. The average number of occlusion-related utterance is 0.67 (SD = 0.58) per task in T-Maze and 0 (SD = 0) in E-Block. When playing T-Maze, even being told that they shall not put their hands on the block, children still did it unconsciously. Such occlusion-related behaviors might cause the inaccurate feedback. When children saw the result on the screen being different from what they expected, they were confused and even changed the right program. Some of them asked in T-Maze: "Why cannot I just tell the computer what I have placed?" Such problem disappeared in test of E-Block for the different way of block recognition. In this case, the occlusion problem of camera-version programming tool is solved by SCM technology [25].

5.3.4. Feedback. In T-Maze the only feedback is the smiling/sad face on the top left corner of screen. While in E-Block, besides that feedback, the LED on wrong programming block in the sequence will also start to flash if a programming error is detected. The total number of feedback-related behaviors and utterances is close to the number of programming operations. We found that in most cases, children would look for feedback after a programming operation unless they were very confident. The average number of feedback-related behaviors is 10 (SD = 1.27) per task in T-Maze and 9 (SD = 1.03) in E-Block. The average number of feedback-related utterances is 0.64 (SD = 0.58) per task in T-Maze and 0.61 (SD = 0.55) in E-Block. The numbers of such behavior and utterance in E-Block differed slightly from that of T-Maze; however, from the collected questionnaire, 9 children said that the feedback on block helped them to find the wrong block much more quickly and accurately compared with that in T-Maze. Only 2 boys did not pay much attention to the block's feedback, because they learned E-Block in the shortest time and always placed the right block. This feedback mechanism can also act as a trial to realize information mutualization between children and the physical objects they manipulate. It can be easily realized by SCM-version programming tool thanks to the inner circuit of each block.

6. Discussion

Figure 10 shows that children devoted a much larger portion of their total behaviors to programming in E-Block (53.0% compared with 36.4%). In addition, Figure 11 shows that the utterances concerning spatial constraint and occlusion problem in E-Block were much fewer than those in T-Maze (9.2% compared with 51.0%). In E-Block, children had fewer behaviors and utterances irrelevant to programming, by which we conclude that E-Block enables children to better focus on programming rather than solving the problem caused by camera. The questionnaires further support our conclusion that 5 out of 11 children said that spatial constraint made it difficult for them to program and 9 out of 11 children said T-Maze is unstable which we attribute to the occlusion problem. After the user study, we found two problems: (a) recognition speed has a crucial influence on children's learning of E-Block, and (b) feedback should be properly designed for children's attentions.

6.1. Recognition Speed. In the user study, a large number of tested children responded that T-Maze is too slow (about 3 seconds to react). They doubted whether their programs are right and can function because of the delay. Such problem, though few, also appeared in the first version of E-Block. The improved E-Block better solved the delay problem by reprogramming each block and changing the communication method between master SCM (in wireless box) and servant SCM (in each programming block). The limit test showed the improved E-Block reacted to the change of block sequence within 0.3 second.

6.2. Feedback. Flashing of LED was inconspicuous based on our user study. Therefore, we reprogrammed and added blue and red LEDs to each block. After the user study, we invited 5 children playing the same tasks as in the user study. We focused on how the improved E-Block solved the problems above and how the improvement influenced children's programming performances.

We video-recorded the programming stage and counted the number of each kind of behaviors and utterances just like the user study. The feedback-related behaviors and utterances were further divided into *feedback on block related* and *feedback on screen related*. The data shows that when playing with improved E-Block, children had a higher proportion of behaviors and utterances related to the feedbacks on block rather than on screen (64% and 72% compared with 43% and 47%, resp.). The user study demonstrated that the improved E-Block reacted faster and provided conspicuous feedback.

7. Conclusion and Future Work

This paper's contribution could be summarized into the following points. Firstly, it presents a programming tool which enables children to write programs by placing wooden blocks. Secondly, it provides a new map method between the block's function and the program semantics. Thirdly, the system offers help on debug by giving feedbacks on both screen and the programming blocks, which effectively helps children to learn programming. We conducted the user experiments on this programming tool. The user study is designed to compare E-Block with the former work: T-Maze. The comparison leads to a further discussion between SCM-version programming tool and camera-version tool. The result shows that E-Block is not only interesting to children but also easy to learn and use. Compared with T-Maze, E-Block better focuses children on programming because of the high proportion of programming operations and low proportion of space-related and occlusion-related utterances. The conclusion is also supported by the result of the interview.

In the future, we intend to design more programming blocks and add more scenarios into this tool to introduce more programming concepts. Feedback on physical object

itself is a great source of inspiration for future work. Future work will mainly focus on getting rid of the computer screen. Children can place the blocks to programming and the operation's result of the program will be shown on the blocks themselves.

Acknowledgments

This research is supported by the National Natural Science Foundation of China under Grants nos. 60970090 and 61272325, the Major State Basic Research Development Program of China (973 Program) under Grant no. 2013CB328805, the Frontier Project of the Knowledge Innovation of Chinese Academy of Sciences under Grant no. ISCAS 2009-QY03, the Cooperation Projects of Guangdong Province and Chinese Academy of Sciences under Grant no. 2011B090300086, and the Cooperation Projects of Chinese Academy of Sciences and Foshan city under Grant No. 2012YS04. The authors would like to acknowledge the support of Tianyuan Gu, Cheng Zhang, Liang He, Tingting Wang, Li Shen, and Muyan Li. We thank the teachers and students in Kindergarten of Chinese Academy of Sciences for their cooperation.

References

[1] S. Papert, *Mindstorms: Children, Computers, and Powerful Ideas*, Basic Books, New York, NY, USA, 1993.

[2] M. Resnick, F. Martin, R. Sargent, and B. Silverman, "Programmable bricks: toys to think with," *IBM Systems Journal*, vol. 35, no. 3-4, pp. 443–452, 1996.

[3] D. Clements, "The future of educational computing research: the case of computer programming," *Information Technology in Childhood Education Annual*, no. 1, pp. 147–179, 1999.

[4] G. Revelle, O. Zuckerman, A. Druin, and M. Bolas, "Tangible user interfaces for children," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI '05)*, pp. 2051–2052, ACM Press.

[5] S. Y. Chen, "Active vision for robotic manipulation," *Industrial Robot*, vol. 39, no. 2, pp. 111–112, 2012.

[6] M. Eisenberg, N. Elumeze, M. MacFerrin, and L. Buechley, "Children's programming, reconsidered: settings, stuff, and surfaces," in *Proceedings of the 8th International Conference on Interaction Design and Children (IDC '09)*, pp. 1–8, June 2009.

[7] A. Begel, *LogoBlocks: A Graphical Programming Language for Interacting With the World*, MIT Press, Boston, Mass, USA, 1996.

[8] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The scratch programming language and environment," *ACM Transactions on Computing Education*, vol. 10, no. 4, p. 16, 2010.

[9] J. A. Fails, A. Druin, M. L. Guha, G. Chipman, S. Simms, and W. Churaman, "Child's play: a comparison of desktop and physical interactive environments," in *Proceedings of the Conference on Interaction Design and Children (IDC '05)*, pp. 48–55, ACM Press, 2005.

[10] P. H. Juan, B. B. Benjamin, D. Allison, and Guimbretiére, "Differences in pointing task performance between preschool children and adults using mice," *ACM Transactions on Computer-Human Interaction*, vol. 11, no. 4, pp. 357–368, 2004.

[11] H. Ishii and B. Ullmer, "Tangible bits: towards seamless interfaces between people, bits and atoms," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI '97)*, pp. 234–241, March 1997.

[12] M. Andrew and O. M. Claire, "Tangibles for learning: a representational analysis of physical manipulation," *Personal and Ubiquitous Computing*, vol. 16, no. 4, pp. 405–419, 2012.

[13] P. Lu, S. Chen, and Y. Zheng, "Artificial intelligence in civil engineering," *Mathematical Problems in Engineering*, vol. 2013, Article ID 145974, 22 pages, 2013.

[14] E. Hornecker and A. Dünser, "Of pages and paddles: children's expectations and mistaken interactions with physical-digital tools," *Interacting with Computers*, vol. 21, no. 1-2, pp. 95–107, 2009.

[15] T. S. McNerney, *Tangible programming bricks: an approach to making programming accessible to everyone [M.S. thesis]*, MIT Press, Boston, Mass, USA, 2000.

[16] T. McNerney, "From turtles to tangible programming bricks: explorations in physical language design," *Personal and Ubiquitous Computing*, vol. 8, no. 5, pp. 326–337, 2004.

[17] D. L. Wang, C. Zhang, and H. A. Wang, "T-Maze: A tangible programming tool for children," in *Proceedings of the 10th International Conference on Interaction Design and Children (IDC '11)*, pp. 127–135, June 2011.

[18] H. Suzuki and H. Kato, "Interaction-level support for collaborative learning: AlgoBlock-an open programming language," in *Proceedings of the 1st International Conference on Computer Support for Collaborative Learning (CSCL '95)*, pp. 349–355, ACM Press, 1995.

[19] M. Sugimoto, T. Fujita, H. Mi, and A. Krzywinski, "RoboTable2 a novel programming environment using physical robots on a tabletop platform," in *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology (ACE '11)*, ACM Press, 2011.

[20] T. S. McNerney, *Tangible programming bricks: an approach to making programming accessible to everyone [M.S. thesis]*, MIT Press, Boston, Mass, USA, 2000.

[21] D. Gallardo, C. F. Juli'a, and S. Jord'a, "TurTan: a Tangible programming language for creative exploration," in *Proceedings of the 3rd annual IEEE international workshop on horizontal human-computer systems (TABLETOP '08)*, pp. 412–420, IEEE Press, 2008.

[22] P. Wyeth and H. C. Purchase, "Tangible programming elements for young children," in *Proceedings of the Extended Abstracts on Human Factors in Computing Systems (CHI '02)*, pp. 774–775, ACM Press, 2002.

[23] P. Wyeth and G. Wyeth, "Electronic blocks: tangible programming elements for preschoolers," in *Proceedings of the International Conference on Human-Computer Interaction (INTERACT '01)*, pp. 496–503, 2001.

[24] H. Shi, W. Wang, N. M. Kwok, and S. Y. Chen, "Game theory for wireless sensor networks: a survey," *Sensors*, vol. 12, no. 7, pp. 9055–9097, 2012.

[25] C. Cattani, S. Chen, and G. Aldashev, "Information and modeling in complexity," *Mathematical Problems in Engineering*, vol. 2012, Article ID 868413, 4 pages, 2012.

[26] E. Schweikardt and M. D. Gross, "RoBlocks: a robotic construction kit for mathematics and science education," in *Proceedings of the 8th International Conference on Multimodal Interfaces (ICMI '06)*, pp. 72–75, November 2006.

[27] M. S. Horn and R. J. K. Jacob, "Tangible programming in the classroom with tern," in *Proceedings of the 25th SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*, pp. 1965–1970, May 2007.

[28] M. S. Horn and R. J. K. Jacob, "Designing tangible programming languages for classroom use," in *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, pp. 159–162, February 2007.

[29] D. L. Wang, Y. Zhang, T. Y. Gu, L. He, and H. A. Wang, "E-Block: a tangible programming tool for children," in *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*, pp. 71–72, ACM Press, 2012.