Interactive Teaching Aids Integrating Building Blocks and Programming Logic

Chien-Hsing Chou[1], Yu-Sheng Su[2], Hui-Ju Chen[1]

[1] Department of Electrical Engineering, Tamkang University, Taiwan
[2] Department of Computer Science and Engineering, National Taiwan Ocean University, Taiwan
chchou@mail.tku.edu.tw, ntoucsiesu@mail.ntou.edu.tw, hueiru4567@gmail.com

Abstract

This study developed interactive teaching aids integrating building blocks and programming logic for children called e-Tuning. The teaching aids comprise two systems: (1) a programming logic board and (2) an e-book and task maps. The design concept of the programming logic board enables children to conduct simple programming without computers. Computer-programming commands are converted into tangible programming blocks easily understood by children. Children can simply place the programming blocks on the programming logic board to complete their programming tasks. The e-book provides assembly methods for Lego robots to complete different programming tasks. Children can follow the assembling steps in the e-book to construct the robot and use it to solve programing tasks. Constructing robots with varied designs enhances the enjoyment of children during the learning process. The e-book also provides diverse story tasks and corresponding task maps. Teachers can apply these story tasks to guide children in using programming blocks to complete the tasks systematically. The results of an experimental course and field trials reveal that through games and tasks, e-Tuning can cultivate basic programming logic, inspire creativity, and develop logical thinking.

Keywords: Tangible user interface, Programming logic, Building block, e-Tuning, Robot

1 Introduction

Because of advancements in technology, learning programming languages has become crucial. Numerous countries have begun encouraging the teaching of programming languages to children. Clement [1] reported that learning programming enhanced children's cognitive abilities for numbers, visual materials, and memory. Fessakis et al. [2] revealed that learning programming languages benefited children's abilities of logical thinking, problem-solving, and communication. However, common programming languages, such as C or Java, contain complicated commands; consequently, they are not suitable tools for children to learn programming. Hence, several researchers have focused on this research topic and developed different visual programming languages as tools for children to study programming [3-7]. Weintrop and Wilensky [3] revealed that visual programming languages have simple interfaces and therefore are suitable tools for children to learn programming. Scratch is a famous programming learning tool designed by the Resnick research group at the MIT Media Lab [4]; it is suitable for children aged 8-16 years, peaking at 12 years. This programing language provides abundant graphical programming blocks with distinct colors and labels for various functions. Children can use the blocks to control characters and program them to perform actions such as talk, move, and jump. They can also design scenes and stages in their programs.

Similarly, Hopscotch [5] is a programming language suitable for children aged 8-12 years. Children can adopt different graphical programming blocks to design a complete game and animation. Compared with Scratch, Hopscotch involves more-diverse programming blocks. Cargo-Bot [6] is another programming logic training app. Through this app, a user can edit consecutive command instructions to control a robotic arm (e.g., moving left or right and using conditional statements) to place boxes at assigned locations in the game. Because of the high difficulty of the tasks, this app is more suitable for senior students. Code.org [7] is an online learning platform that provides programming tasks of various difficulty levels for children to learn programming logic. Although these visual programing tools are simple and convenient for learning programing, the teaching themes may be less entertaining; therefore, children may easily become bored and lack motivation to continue learning.

Although visual programming languages have been commonly employed in teaching programming to children, computers or tablets maybe not be suitable for teaching programming to young children [8-9].

*Corresponding Author: Yu-Sheng Su; E-mail: ntoucsiesu@mail.ntou.edu.tw
DOI: 10.3966/160792642019102006003

Sewunet et al. [8] maintained that extended use of computers or tablets could affect children's eye development. Stowell's research [9] also indicated that using mobile devices as learning tools may not produce favorable learning results. Most children view mobile devices as a platform for gaming and social networking. Therefore, they are easily distracted by gaming apps or video software, preventing them from concentrating on learning. Moreover, the aforementioned programming software packages are excessively complicated for young children. Therefore, some researchers have designed learning aids for enabling preschool children to learn programming.

In recent years, tangible user interface (TUI) has emerged as a new type of user interface, connecting the digital and physical worlds [10-13]. TUIs are employed to improve existing learning tasks. With TUIs, children can interact with the learning tasks by selecting and positioning physical objects. TUIs employ physical actions to assist learning tasks, and tangible objects provide diverse opportunities for thinking about the world. This learning strategy is effective and conducive to children's learning. Merrill et al. applied technology and methodology from wireless sensor networks [14] to develop Siftables, which are compact devices with sensing, graphical display, and wireless communication capabilities. Users can physically manipulate and assemble Siftables as a group to interact with digital information and media. Farr et al. [15] applied a construction toy (Topobo) to promote social interaction in autistic children. Girouard et al. [16] developed Smart Blocks, which are augmented mathematical devices enabling users to explore the concepts of volume and surface area of three-dimensional objects. Their interface supports physical manipulation to explore spatial relationships and provides continuous feedback.

Many researchers have recently employed robots to develop an appropriate programming teaching aid for children. Cubetto [17] is a programming learning aid for preschool children. Children use tangible action blocks (such as blocks instructing a robot to move left or right) to control the movement of a two-wheeled robot. Martinez et al. [18] experimented with a school intervention to teach fundamental programming logic concepts to 3-11-year-old students with a robot-programming platform. The experimental results showed that all students could intuitively learn sequences, conditional statements, loops, and parameters. By adopting the design concepts of previous research, this study designed interactive programming teaching aids called e-Tuning for children. As shown in Figure 1, e-Tuning contains a tangible graphical programming interface, a Lego robot, an e-book, and a story task map. The robot control program was converted into tangible programming blocks. In the proposed system, children are not required to use computers or tablets. Instead, through TUI, they can complete the robot control program by arranging programming blocks on a programming logic board. Compared with other researches [17-18] our programming blocks also include light, sound, and button sensors that enable children to learn the concept of conditional statements in programming.

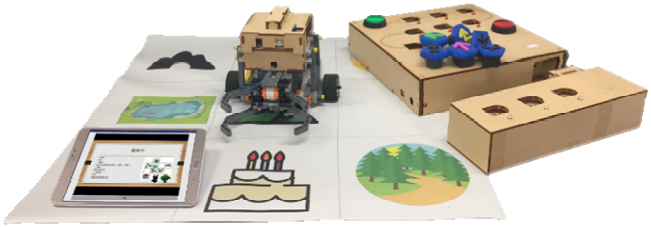


Figure 1. Proposed e-Tuning system

To increase children's learning motivation, e-Tuning provides numerous programming blocks and story tasks as teaching content. Our e-book (a story task question bank) includes programming learning materials of different difficulty levels. Teachers can use strategies of storytelling and task completion to attract children to cultivate basic programming logic. Furthermore, e-Tuning allows children to construct robots for completing different story tasks by assembling Lego bricks. The results of the field test reveal that children enjoy using their own robots to complete story tasks. This practice further enhances children's learning motivation.

2 E-Tuning

This study designed a set of interactive programming teaching aids for children called e-Tuning. Figure 2 shows a flowchart of the e-Tuning process. Our e-Tuning adopts a tangible programming logic board and game-based learning to teach students programming logic. To control the robot's movement, children must only place programming blocks on the programming board and press the green button to execute their program. The story task bank includes several story tasks with difficulty levels ranging from easy to difficult. Children can choose a story task from the e-book suitable to their abilities. To help children study programming logic, various restrictions are implemented in corresponding story tasks. If children successfully complete a story task, they can select a new story task from the task bank. Alternatively, they can repeatedly adjust their programming blocks until they achieve the goal of the story task.

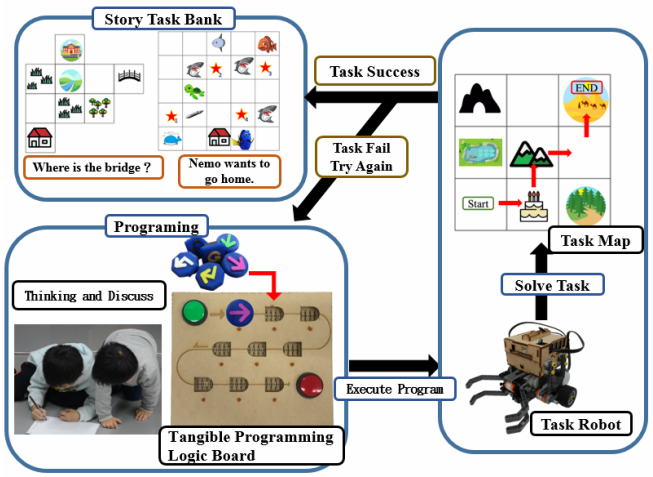
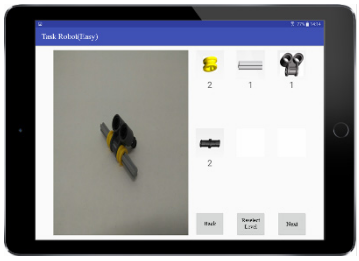


Figure 2. Flowchart of e-Tuning

3 Construction of Task Robot

Research indicates that building with Lego bricks increases children's spatial cognition [19]. To help students understand the structures of gears and robots, we teach children to use Lego bricks to build task robots. First, children choose a story task from the story task bank in the e-book and assemble the robot for the story task. The assembling steps for the task robot are also provided by the e-book. Figure 3(a) shows a step to assemble the task robot. In each assembling step, the brick types and their numbers are listed, and the assembly result is also provided. Following the assembling steps, children build their own robots to complete this story task. Figure 3(b) shows a type of robot gripper used in the study.

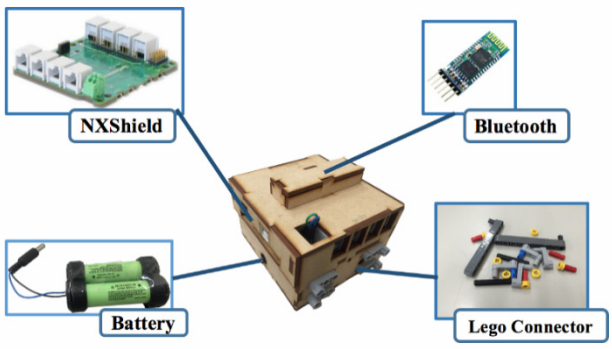


(a) Example of a brick assembling step

(b) Robot gripper

Figure 3. A type of robot gripper used in the study

To control the robot's movement using the programming logic board, we designed a new control core, rather than using the Lego Mindstorms NXT controller [20]. Figure 4(a) shows the control core. The control core uses an Arduino UNO and an NXShield as its embedded system responsible for I/O control and communication. The embedded system handles three functions: (a) communication with the programming logic board through a Bluetooth module; (b) power to the motor; and (c) detection of the states of button, light, and sound sensors. Figure 4(b) shows the NXT motor and three sensors applied in this study.



(a) Control core

(b) NXT motor and sensors

Figure 4.

4 Tangible Programming Interface

A tangible programming interface is an easily comprehensible and user-friendly system that enable preschool children to learn programming. However, using a computer to learn programming remains unsuitable for younger children. Manches and Price [21] indicated that tangible learning interfaces (i.e., noncomputer) benefit children's memory and comprehension of graphs and colors. Therefore, this study integrated the concept of blocks in the graphical programming interface. The programming interface comprised three components: programming blocks, a programming logic board, and a digital logic board app. Additional details are presented in the following sections.

4.1 Programing Blocks

To control the robot, we converted complex programing codes into tangible programming blocks. As shown in Figure 5, the left part is the programming code for controlling the robot to move forward; however, this study designed a tangible program block to represent the programming code. Children can control the robot to move forward by placing this programming block on the programming logic board.

```
void TWO_servo_Forward_a(){

  nxshield.bank_b.motorRunDegrees(SH_Motor_2,
                    SH_Direction_Forward,
                    50,
                    540,
                    SH_Completion_Dont_Wait,
                     SH_Next_Action_BrakeHold);
  nxshield.bank_b.motorRunDegrees(SH_Motor_1,
                    SH_Direction_Forward,
                    50,
                    540,
                    SH_Completion_Wait_For,
                    SH_Next_Action_Brake);

  BTSerial.write('Y');
  nxshield.bank_a.motorReset();
  nxshield.bank_b.motorReset();
  }
```

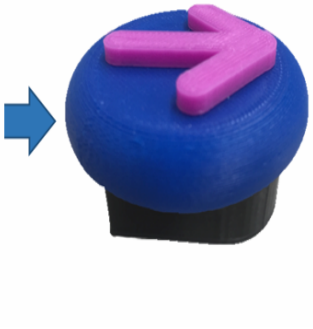


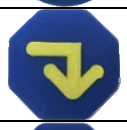
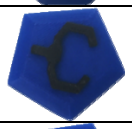
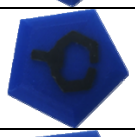
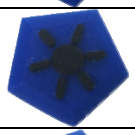
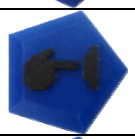
Figure 5. Converting complex programming code into a tangible programming block

To control the robot's movement, this study designed ten types of programming blocks. Table 1 presents the correspondence between the programming blocks and robot movements, usages, and purposes. The blocks are classified into shapes and colors to enable children to distinguish them easily. For example, programming blocks for "forward" and "backward" are purple and green, respectively, and are both circular. Programming blocks for "turn right" and "turn left" are yellow and white, respectively, and are both octagonal. Programming blocks representing Lego sensors are all black and pentagonal. They are applied to detect the states of light, sound, and press action. Programming blocks for "open gripper" and "close gripper" are in the same category as the Lego sensors and displayed as black and pentagonal. These two blocks control the robot grasping objects. The program block "subroutine" is brown and square; it represents a subroutine function used to execute three programming blocks on the extensive board.

4.2 Programming Logic Board

In contrast to the conventional programming teaching method, in which each learner uses a computer, the programming logic board (Figure 6) designed in this study does not require computers. Moreover, it enables children to experience the fun of learning programming together through discussion. This board includes nine bullet-shaped grooves for placing programming blocks. It also includes an auxiliary line indicating the program execution sequence. After children place programming blocks, they can press the green button to execute the program. The activation of the LED light under a certain groove signifies that the program is currently executing that groove. When the program reaches an empty groove, the LED light under the groove begins to flash, and the program execution stops.

Table 1. Ten types of programming blocks

Programming Block	Robot Movements and Usages
	Forward
	Backward
	Turn Right
	Turn Left
	Open Gripper
	Close Gripper
	Light Sensor
	Button Sensor
	Sound Sensor
	Subroutine

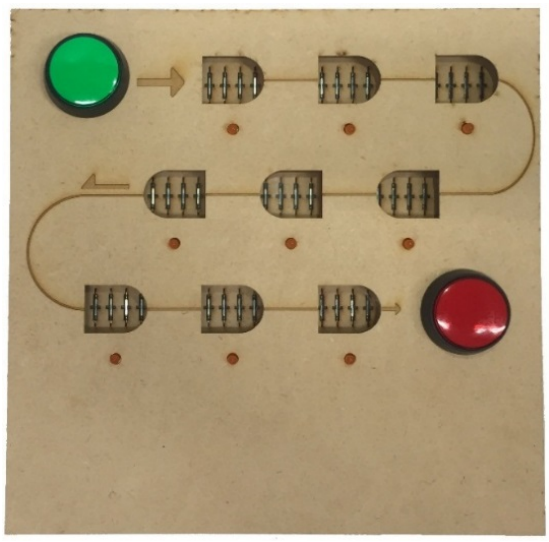


Figure 6. Programming logic board

In addition to the programming logic board, this study designed a subroutine extension board (Figure 7). The concept of the subroutine extension board is the same as that of a subroutine in programming. When the program is executed to a "subroutine" programming block on the programming logic board, the program calls the subroutine function and then executes programming blocks on the extension board before returning to the programming logic board to continue executing the next programming block.

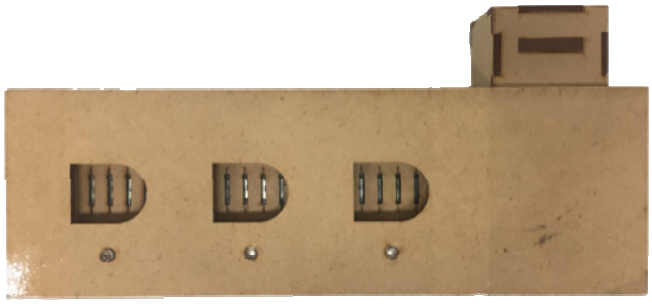


Figure 7. Subroutine extension board

4.3 Digital Logic Board App

We also developed an app called digital logic board. The app operates on tablets and performs the same functions as the aforementioned programming logic board. Figure 8 shows the appearance of digital logic board. This app contains includes 21 bullet-shaped grooves for placing programming blocks, and five bullet-shaped grooves for subroutine functions. The app lets children place more programming blocks than the programming logic board when they want to deal with a difficult story task.

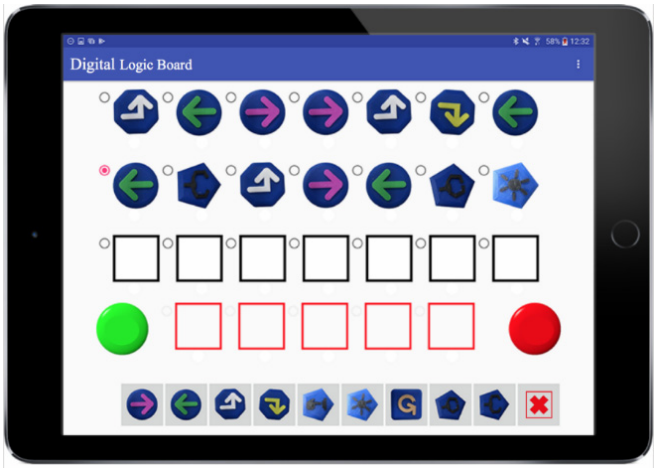


Figure 8. Digital logic board

5 E-book and Story Tasks

We designed several story tasks with difficulty levels ranging from easy to difficult in the e-book. Children can choose a story task suitable to their ability from the task bank to practice. An example of a story task in the proposed e-book is illustrated in Figure 9. To help children understand programming logic, various restrictions are implemented in some story tasks. For example, the number of programming blocks that may be limited for use, tangible objects (block trees) may be added or sensor use may be required. The various types and difficulty levels of the story tasks enable children set and reach desired learning goals. As shown in Figure 9, children are asked to use the "turn left" programming block only once in this story task. This story task was not designed to have a standard answer. Story tasks may have several answers. Children can use a subroutine to achieve the restriction of this task, or they can separate this task into two subtasks to complete it. In brief, if children achieve the goals, they are considered to have successfully completed the tasks. We observed that when children repeatedly adjust their programming blocks until they achieve the goal of story task, they usually find a superior solution.

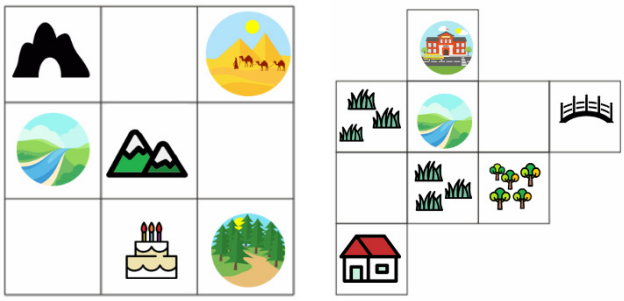
Figure 9. Example of a story task in the proposed e-book

We designed four task maps for the aforementioned story tasks; these maps have different difficulty levels, with the easiest map comprising 3 × 3 cells and the most challenging map comprising 5 × 5 cells. Each story task corresponds to an exclusive task map. In addition to difficulty levels, the four task maps involve various missions. In the basic map, children only learn how to control the robot's movement. In the elementary map, story tasks are solved using sensors or tangible objects. In the intermediate map, the subroutine concept is introduced, and sensors are included to solve the story tasks. In the advanced map, story tasks are solved using the subroutine concept, sensors, tangible objects, or digital logic board app. Table 2 summarizes details of these maps.

Table 2. Details of Four Task Maps

Task Map	Map Size	Subroutine	Sensors	Tangible Objects	Digital Logic Board App
Basic Map	3×3				
Elementary Map	4×4		v	v	
Intermediate Map	5×5	v	v		
Advanced Map	5×5	v	v	v	v

Figure 10(a) and Figure 10(b) shows the basic and elementary maps for story tasks, respectively. The basic map (Figure 10(a)) comprises nine cells; the low number of cells in the map renders it suitable for children to practice elementary programming. According to the story task, children try to move their robot to a different cell by placing programming blocks on programming logic board. In addition to teaching children programming logic, this map teaches children about nature; for instance, a desert is hot, and bears live in mountain caves. Figure 10(b) presents the elementary map. With a low number of cells, this map is designed to help children to learn the concept of sensors while they execute their programs. Lego sensors are added to the task for this map. An example of a story task is as follows: "It is nightfall. The robot wants to find the cat in the grass, but it is too dark. The robot needs the help of a flashlight." In this example, for the robot to navigate the grass, children first set the Lego light sensor on their robot. They then must use the "light sensor" programming block while they execute their programs. When the robot arrives at the grass cell, children shine the flashlight on light the sensor to enable the robot to continue moving.



(a) Basic task map (b) Elementary task map

Figure 10.

Figure 11 shows the intermediate task map. The difficulty level of this map is higher than that of the basic or elementary maps. The goals of the story tasks are more difficult to achieve, therefore the subroutine concept is applied in this map. Furthermore, more programming blocks must be used. The map contains marine organisms and four numbered stars. The stars indicate task instructions that guide children to move the robot to certain cells and presses button sensors to execute the following tasks. Moreover, the inclusion of different types of fish on the map helps children learn about marine organisms. This map is suitable for story tasks requiring the use of sensors.

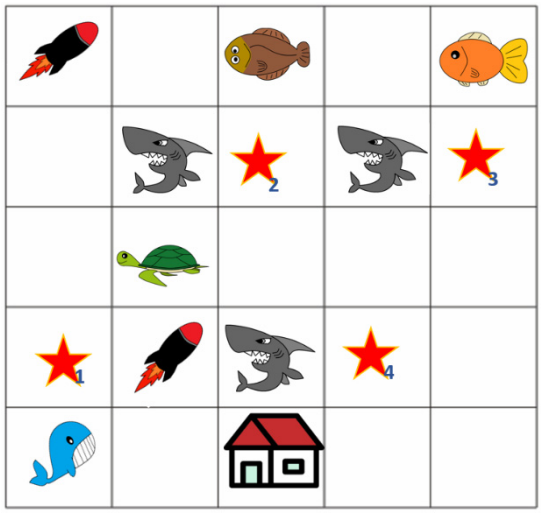


Figure 11. Intermediate task map

Figure 12 depicts the advanced task map. This map contains more features than the maps of other difficulty levels. Because more restrictions are imposed in the story tasks, children must apply more steps or a subroutine to achieve goals. For example, a story task must be completed without the robot passing through certain cells. Moreover, this map allows children to use the digital logic board app; therefore, more programming blocks can be placed simultaneously.

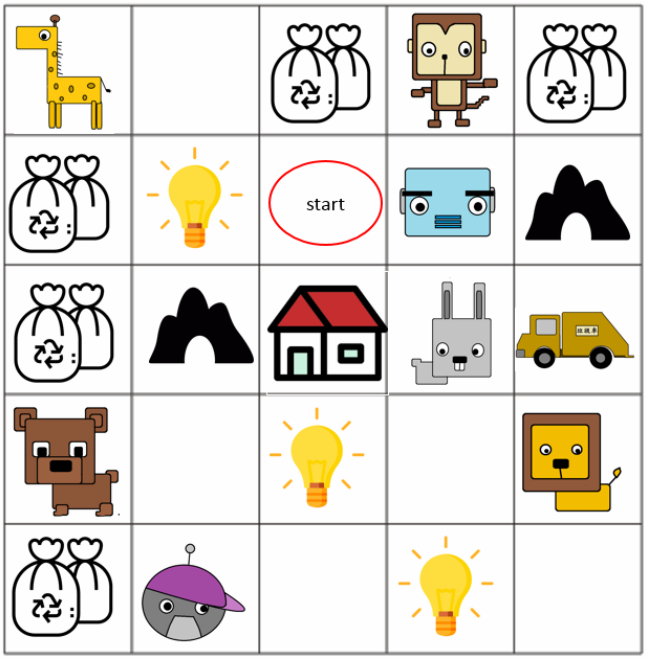
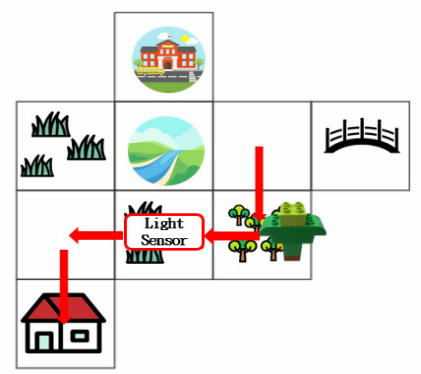


Figure 12. Advanced task map

The following examples demonstrate story tasks at the elementary level (including sensors and tangible objects) and advanced level (including subroutine and limited programing blocks) and their corresponding task maps.

5.1 Example 1: Elementary-Level Task (with Sensor and a Tangible Object)

- Story plot and task: It is late in the evening, and the robot (indicated by the blue dot in Figure 13(a)) must go home. The robot must collect some plants on the way home. However, because of the darkness, the robot needs a flashlight when walking through the grass to find the way home.
- Task description: The blue dot indicates the robot's initial position. The robot must cut down a tree in the woods (i.e., the gripper must be used to grasp a tiny tree) and then go home. When the robot wants to pass through any grass cell, a light sensor and flashlight must be used.
- Solution: As indicated in Figure 13(a), the robot is located at the blank cell. The grippers are employed to collect the tiny tangible tree. The robot then turns right, walks straight, and reaches the grass. The "light sensor" programming block makes the robot stop at the grass cell before executing the next move. Children must apply the flashlight to shine the sensor to pass through the grass cell. The next moves are to walk straight, turn left, and then walk straight to reach home (illustrated by the house symbol). This story task has other solutions; Figure 13(b) provides a solution as a reference.



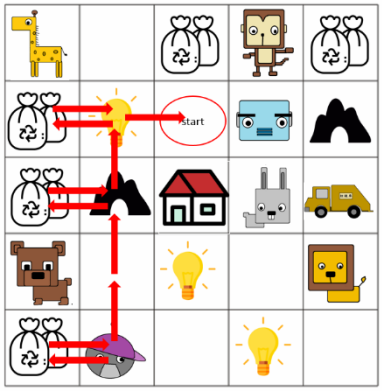
(a) Simulated route of a robot for Example 1

(b) corresponding programming blocks applied to complete this story task

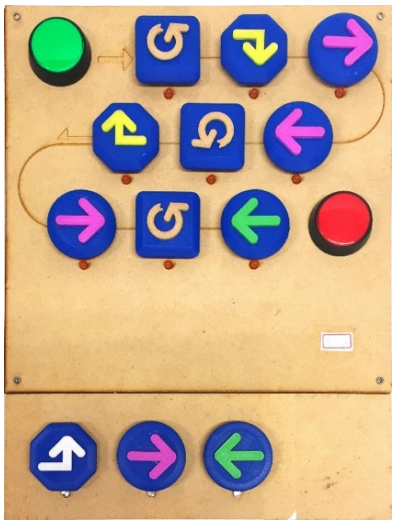
Figure 13.

5.2 Example 2: Advanced-Level Task (with Subroutine)

- Story plot and task: The robot in the purple hat intends to return to the robot headquarters (indicated by "Start" in Figure 14(a)). As instructed by the robot's supervisor, the robot must clean the road on the way back. The robot must return with three bags of waste to be allowed in. Please help the robot.



(a) Simulated route of a robot for Example 2



(b) corresponding programming blocks applied to complete this story task

Figure 14.

- Task description: The restrictions of this task are that the "turn left" programming block may be used only once, and on the way to the "Start" cell, the robot must pass through three cells with waste illustrations.
- Solution: As shown in Figure 14(a), the robot is located at the cell showing a robot wearing a purple hat. Because three bags of waste must be collected and a "turn left" programming block may be used only once, a subroutine must be employed to complete this task. Programming blocks "turn left," "forward," and "backward" can be used in the subroutine. The robot must first execute a subroutine

to enter the bag cell then turn right and proceed forward across two cells to reach the cell showing a mountain cave. Subsequently, the robot must use the subroutine again to enter another bag cell, and return to the mountain cave. The robot must then turn right, move forward, execute the subroutine again, and proceed backward to reach the "Start" position. Because the subroutine concept is difficult, children may use the digital logic board app to place multiple programming blocks to solve the task, as demonstrated in Figure 15. Other solutions are available for completing the task; Figure 14(b) and Figure 15 provide two of them as references.

Figure 15. Corresponding programming blocks for solving Example 2 using the digital logic board app

6 Experimental Course and Field Trials

To test whether e-Tuning benefits teaching children programming logic, this study designed a 3-day experimental course. Figure 16 shows photos of the programming logic training course. We examined whether participants' concepts regarding programming logic had improved after the 3-day course. The participants were six children aged 5-10 years that were divided into three groups. The first and second groups comprised one fourth-grader and one second-grader each, whereas the third group comprised one first-grader and one student in the second year of preschool. The teaching plan is detailed in Appendix A. First, students were taught to construct their own task robots with Lego bricks. The concept of the programming logic board and its usage methods were introduced to the students. By playing games, the students learned how to control their robots with the programming logic board and practiced the beginning task on the basic task map.

After the students had familiarized themselves with how to control the robot's movement, the advanced tasks with sensors (light or sound sensor) were added, and the students were taught how to use the sensors and their functions. For example, the robot lost in the grass requires light to find the correct route. In this instance, the students must apply the flashlight to shine the light sensor to enable the robot to execute the following program. By contrast, if children must repeat a subtask several times, such as moving forward and grasping, the teacher can guide students to adopt a subroutine function for this subtask.

(a) Constructing their own task robots

(b) First group

(c) Second group

(d) Third group

Figure 16. Programming logic training course

During the training course, each group completed tasks on four task maps of various difficulty levels. Because the course entailed conducting the tasks in teams, students were more willing perform the tasks themselves, ask teachers questions, and discuss with one another during their learning process. The solved story tasks were never repeated. Each group solved 11-13 story tasks. The details of solving tasks are recorded in Appendix B. Tables 3-4 briefly presents the average times for the three groups. Groups 1 and 2 comprised two older children (8 and 10 years); group 3 comprised two younger children (5 and 7 years). After the 3-day training course, we made the following observations:

(1) Students enjoyed using the robot they constructed to perform tasks. Therefore, allowing students to construct their own robots can increase their learning motivation. Moreover, during the building process, the students learned the function of gears and a robot's structural design.

(2) Although each group was assigned different story tasks, we found that students engaged in intergroup discussions, helping one another and shortening the time they spent on solving story tasks.

(3) Table 2 and Appendix B show that groups 1 and 2 (older children) solved story tasks faster than group 3 (younger children) on the basic and elementary task

maps. However, after younger students familiarized themselves with the programming logic board, they could perform as well as the older children.

(4) During 3 days of practice, the time required by each group to solve story tasks decreased.

(5) After solving many story tasks, the students gained a clearer understanding of command-by-command execution in computer programs and of the actions executed by the robot.

Table 3. Average spending time of three groups for solving story task (Basic and Elementary Maps and Intermediate Map)

	Basic and Elementary Maps		Intermediate Map	
	# of Tasks	Average Time	# of Tasks	Average Time
Group 1	4	266	4	549
Group 2	4	247	5	396
Group 3	4	387	4	516

Table 4. Average spending time of three groups for solving story task (Advanced Map)

	Advanced Map	
	# of Tasks	# of Tasks
Group 1	3	723
Group 2	4	447
Group 3	3	627

To further test e-Tuning, field trials were performed with 29 children aged 5-12 years (Figure 17). This was the first time to use e-Tuning for all these children. We introduced the concept of the programming logic board and its usage methods and let them practice the beginning task on the basic map. After the children familiarized themselves with how to control robot's movement, a programming test "maze" was performed. The children are asked to control the robot to pass the maze. Appendix C details the test results. For comparison, the same programming test was performed by the students who participated in the 3-day course of programming logic training. Table 5 presents the average and fail times of two groups solving the maze test. Students who participated in the programming course required less time to solve the test without fail. This comparison verifies the benefit of our system and programming logic training course.

Figure 17. Field trials of e-Tuning

Table 5. Average spending time and fail times of two groups for solving maze test

	Average Spending Time (s)	Average Fail Frequency
Children without Participated in Programming Course	326	2.2
Children with Participated in Programming Course	202	0

7 Conclusions

This study developed interactive teaching aids integrating building blocks and programming logic for children. Instead of computers, the proposed e-Tuning adopts a tangible programming logic board and game-based learning to teach students programming logic and train their logical thinking abilities while increasing learning motivation. Computer-programming commands are converted into tangible programming blocks easily understood by children. Children can simply place the programming blocks on the programming logic board to complete their programming tasks. To increase children's learning motivation, e-Tuning provides numerous programming blocks and story tasks as teaching content. Teachers can use strategies of storytelling and task completion to attract children to cultivate basic programming logic. The results of an experimental course and field trials reveal that through games and tasks, e-Tuning can cultivate basic programming logic, inspire creativity, and develop logical thinking.

Acknowledgements

This study was supported by the Ministry of Science and Technology (MOST), Taiwan, R.O.C., under MOST 108-2511-H-019-002, MOST 108-2511-H-019-003, MOST 107-2511-H-008-007, MOST 107-2511-H-019-003, and MOST 107-2221-E-032-042.

References

[1] D. H. Clements, Young Children and Technology, *Early Childhood Science, Mathematics, and Technology Education*, Washington, DC, 1999, pp. 92-105.

[2] G. Fessakis, E. Gouli, E. Mavroudi, Problem Solving by 5-6 Years Old Kindergarten Children in a Computer Programming Environment: A Case Study, *Computers & Education*, No. 63, pp. 87-97, April, 2013.

[3] D. Weintrop, U. Wilensky, To Block or Not to Block, That is the Question: Students' Perceptions of Blocks-based Programming, *Interaction Design and Children of the 14th International Conference*, Boston, MA, 2015, pp. 199-208.

[4] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, Y. Kafai, Scratch: Programming for All, *Association for Computing Machinery*, Vol. 52, No. 11, pp. 60-67, November, 2009.

[5] Hopscotch, https://www.gethopscotch.com/.

[6] Cargo-bot, https://itunes.apple.com/tw/app/cargobot/id519690804?mt=8.

[7] Code.org, https://code.org/.

[8] S. A. Sewunet, K. K. Aredo, M. Gedefew, Uncorrected Refractive Error and Associated Factors among Primary School Children in Debre Markos District, Northwest Ethiopia, *BMC Ophthalmology*, No.14, pp. 95, July, 2014.

[9] J. R. Stowell, Use of Clickers vs. Mobile Devices for Classroom Polling, *Computers & Education*, Vol. 82, pp. 329-334, March, 2015.

[10] B. Schneider, J. Wallace, P. Blikstein, R. Pea, Preparing for Future Learning with a Tangible User Interface: The Case of Neuroscience, *IEEE Trans. on Learning Technologies*, Vol. 6, No. 2, pp. 117-128, April-June, 2013.

[11] D. Xu, Tangible User Interface for Children: An Overview, *Department of Computing, University of Central Lancashire*, Preston, UK, 2005.

[12] O. Shaer, E. Hornecker, Tangible User Interfaces: Past, Present, and Future Directions, *Foundations and Trends in Human-Computer Interaction*, Vol. 3 No.1-2, pp.1-137, March, 2010.

[13] M. S. Markova, S. Wilson, S. Stumpf, Tangible User Interfaces for Learning, *International Journal of Technology Enhanced Learning*, Vol. 4, No. 3/4, pp. 139-155, January, 2012.

[14] D. Merrill, J. Kalanithi, P. Maes, Siftables: Towards Sensor Network User Interfaces, *Tangible and Embedded Interaction of the 1st International Conference*, Baton Rouge, Louisiana, 2007, pp. 75-78.

[15] W. Farr, N. Yuill, H. Raffle, Social Benefits of a Tangible User Interface for Children with Autistic Spectrum Conditions, *Autism*, Vol. 14, No. 3, pp. 237-252, May, 2010.

[16] A. Girouard, E. T. Solovey, L. M. Hirshfield, S. Ecott, O. Shaer, R. J. K. Jacob, Smart Blocks: A Tangible Mathematical Manipulative, *Tangible and Embedded Interaction of 1st International Conference*, Baton Rouge, Louisiana, 2007, pp. 183-186.

[17] Cubetto, http://makezine.com/2013/12/04/teachingprogram-ming-to-children/.

[18] C. Martinez, M. J. Gomez, L. Benotti, A Comparison of Preschool and Elementary School Children Learning Computer Science Concepts through a Multilanguage Robot Programming Platform, *Innovation and Technology in Computer Science Education of ACM Conference*, Vilnius, Lithuania, 2015, pp. 159-164.

[19] A. James, Learning in Three Dimensions: Using Lego Serious Play for Creative and Critical Reflection across Time and Space, in: P. Lake & P. Layne (Eds.), *Global Innovation of Learning and Teaching: Transgressing Boundaries*, Springer, New York, 2014, pp. 275-294.

[20] Mindstorms NXT, http://www.class roomantics.com/2014/10/29/great-deal-on-lego-mindstor ms-nxt/lego-nxt-education-9797-inside-box/.

[21] A. Manches, S. Price, Designing Learning Representations around Physical Manipulation: Hands and Objects, *Interaction Design and Children of the 10th International Conference*, Ann Arbor, USA, 2011, pp. 81-89.

Biographies

Chien-Hsing Chou received the B.S. and M.S. degrees from the Department of Electrical Engineering, Tamkang University, Taiwan, in 1997 and 1999, respectively, and the Ph.D. degree at the Department of Electrical Engineering from Tamkang University, Taiwan, in 2003. He is currently an assistant professor of electrical engineering at Tamkang University, Taiwan. His research interests include machine learning, interactive learning, image analysis and recognition, human computer interaction.

Yu-Sheng Su received the Ph.D. degree from Department of Computer Science and Information Engineering, National Central University, Taiwan, in 2010. He is currently an assistant professor of computer science and engineering at National Taiwan Ocean University, Taiwan. His interests include social media mining, AIoT, cloud computing, and computational thinking.

Hui-Ju Chen received the B.S. degrees from Department of Biomedical Engineering, I-Shou University, Taiwan, in 2014 and the M.S. degrees from the Department of Electrical Engineering, Tamkang University, Taiwan, in 2016. She is currently in the Ph.D. program of electrical engineering at Tamkang University, Taiwan. Her research include human computer interaction, interactive learning and mechanism design.

Appendix A

Table A1. Teaching plan for programming logic training.

Course Name	Interactive experimental course on programming logic training
Teaching Theme(s)	Programming logic training
Sources of Teaching Material	Self-compiled
Teaching Aids	Lego Mindstorms NXT, Lego bricks, programming logic board, e-book, digital logic board app, iPad, task maps
Teaching Objects	Six children aged 5-10 years
Teaching Duration	7 hours per day for 3 days
Teaching Objectives	Instead of computers, this course adopts a tangible programming logic board and game-based learning to teach students programming logic and train their logical thinking abilities while increasing learning motivation.
Course Content	**Teaching theme 1:** Teach students to use Lego bricks to assemble task robots. Help students understand the structures of gears and robots. **Teaching theme 2:** Teach students to control the robot's movement by placing programming blocks on the programming logic board. Students practice with various story tasks on the corresponding task maps. **Teaching theme 3:** Add sensors to robots. Teach students the function of sensors. Help the students use the corresponding programming blocks for sensors to complete the story tasks. **Teaching theme 4:** Teach students the concept of subroutines. Guild students use programming blocks to create subroutines to complete the corresponding story task.
Assessment Method	Divide students into three groups (each with two students), and instruct them to complete multiple story tasks. The assessment method is outlined as follows: 1. Allow students to solve different story tasks (with or without sensors; with or without subroutines) using the programming logic board, e-book, digital logic board app, and their robots. Record the time required for solving each story task. 2. Observe students' performance during the learning and task-solving processes.

Appendix B

Table B1. Spending Time for Basic and Elementary Maps.

	Task ID	With Sensor	Spending Time(s)	Average Time(s)
Group 1	1	V	485	266
	2		265	
	3		218	
	4		96	
Group 2	1	V	405	247
	2		203	
	3		213	
	4		166	
Group 3	1	V	720	387
	2		404	
	3		207	
	4		215	

Table B2. Spending Time for Intermediate Map

	Task ID	With Sensor	Spending Time(s)	Average Time(s)
Group 1	1	V	535	549
	2	V	611	
	3		425	
	4		626	
Group 2	1		405	396
	2	V	548	
	3	V	311	
	4		407	
	5	V	317	
Group 3	1	V	555	516
	2		541	
	3		456	
	4		510	

Table B3. Spending Time for Advanced Map

	Task ID	With Sensor	Spending Time(s)	Average Time(s)
Group 1	1	V	845	723
	2	V	705	
	3	V	620	
Group 2	1	V	420	447
	2	V	452	
	3	V	440	
	4		477	
Group 3	1	V	747	627
	2		666	
	3	V	468	

Appendix C

Table C1. Spending time and fail times of 29 children for solving maze test.

Children ID	Age (5-12)	Spending Time(s)	Fail Frequency
1	8	366	4
2	5	468	3
3	7	349	3
4	6	397	3
5	5	455	5
6	5	303	1
7	8	340	3
8	5	530	0
9	7	364	4
10	5	390	2
11	6	579	4
12	6	474	2
13	10	210	1
14	9	330	5
15	9	440	2
16	12	224	0
17	10	322	2
18	10	244	2
19	12	158	0
20	10	185	1
21	10	270	2
22	11	211	2
23	11	185	2
24	10	233	2
25	10	276	2
26	10	313	2
27	12	213	1
28	11	326	2
29	9	300	2