The Effects of a Robot Game Environment on Computer Programming Education for Elementary School Students

Jaekwoun Shim, Daiyoung Kwon, and Wongyu Lee

***Abstract*—In the past, computer programming was perceived as a task only carried out by computer scientists; in the 21st century, however, computer programming is viewed as a critical and necessary skill that everyone should learn. In order to improve teaching of problem-solving abilities in a computing environment, extensive research is being done on teaching–learning methods, types of teaching software, the educational environment, and related tools. This paper, based on diverse experimental results, proposes an environment where elementary students can easily learn and practice computer programming. The proposed robot game environment used a tangible programming tool with which students can easily create robot programs, without learning syntax, and then validate their programming results; it can also provide various game activities to incite students' interest. Observation of elementary school students placed in the robot game environment confirmed the tool's usability and entertainment aspects, and students' attitudes toward programming and their understanding of programming concepts improved.**

***Index Terms*—Computer engineering, elementary school, games, problem solving.**

I. Introduction

THE CURRENT digital age has created an information society where it is normal to use digital devices for information processing activities [1]. Nowadays, programming for information processing is no longer just work for engineers and scientists, but has become a common activity for almost anyone [2], [3]. Computer programming is a valuable educational activity that not only helps students improve their IT-specific problem-solving abilities, but can also enhance their critical thinking [4], [5]. Therefore, diverse curricula for programming education have been proposed for K–12 students [6]–[8], but such educational approaches demand much effort because programming activities are very difficult for K–12 students [9]. Most students who have learned programming think that programming is very difficult. In a computer programming class, students must understand fundamental programming concepts (such as sequence, repetition, condition and branch, and function and variable), and simultaneously learn the use of syntax and instructions for programming languages and tools [10]. Accordingly, students generally carry a heavy cognitive load during the programming learning process [11]. Programming is even more difficult for students for whom English is not a native language because most programming tools use English. Consequently, students develop negative perceptions of programming. Particularly for female and younger students, motivation for, and satisfaction with, learning programming decreases [12].

Educational programming languages (EPLs), such as Scratch and E-toys, have been developed for K–12 students to mitigate the various difficulties of programming education [13]. These EPLs use a tile scripting based on a graphic user interface as a coding method, instead of typing text commands. This makes the instructions easy to learn and helps students reduce syntax errors. Students can understand programming concepts and procedures intuitively because the EPL's instructions are mostly about handling visual and aural objects, and the programming results are developed as a type of animation or simulation [14]. Recently, tangible user interfaces (TUIs) have been developed to provide more intuitive interactions in computing environments [15], [16]. TUI programming tools help young novice programmers complete programming activities without the burden of learning tool usages by using intuitive handling, such as connecting and stacking command objects [17]. These TUI programming tools can be beneficial for learning syntax and commands in English, a difficulty for foreign-language programming learners. Consequently, TUI programming tools are very effective for children's programming education [18], [19].

On the other hand, numerous studies have been carried out in efforts to improve student motivation and interest in programming education. An educational robot is an effective way of doing this because it provides experiences of making and controlling various real-world objects such as an elevator or a car [20]. Students' programming abilities can be improved spontaneously through the experience of controlling and manipulating a robot, so an educational robot is very suitable for programming education [21].

Game activities can also be an effective leaning method for improving students' motivation and interest [22]. A game

Manuscript received July 8, 2015; revised December 3, 2015, May 16, 2016, and October 10, 2016; accepted October 12, 2016. This work was supported by the National Research Foundation of Korea through the Korean Government under Grant NRF-2013R1A2A2A03016926. *(Corresponding author: Wongyu Lee.)*

J. Shim and D. Kwon are with the Department of Computer Science Education, Graduate School, Korea University, Seoul 136-701, South Korea (e-mail: jaekwoun.shim@inc.korea.ac.kr; daiyoung.kwon@inc.korea.ac.kr).

W. Lee is with the Department of Computer Science and Engineering, College of Informatics, Korea University, Seoul 136-701, South Korea (e-mail: lee@inc.korea.ac.kr).

Digital Object Identifier 10.1109/TE.2016.2622227

activity helps students focus and immerse themselves in learning through the process of competition and cooperation, even if the learning contents themselves are not interesting [23]. Additionally, a game activity provides experiences for establishing various problem-solving strategies for winning [24]. Therefore, a game activity can be effective for helping students focus on programming and for improving their fundamental algorithm design abilities, required in learning programming. As a result, a game activity has been evaluated as an appropriate learning method in programming education [7], [25].

The aim of the research presented here is to propose a practical educational programming environment for young students that integrates the effective programming tools, material, and learning methods previously researched. The robot game programming environment was developed to satisfy the following criteria. First, young students should be able to readily understand the programming tools and use them intuitively. Second, it should convincingly arouse students' interest and motivation. Finally, students should be able to concentrate on, and immerse themselves in, the programming process. Consequently, the proposed environment was composed of a tangible programming tool, an educational robot, and game boards. In addition, two robot games were developed for evaluation of the proposed programming environment. Fifty elementary school students participated in the study carried out to measure and analyze the elements of usability, edutainment, programming attitude, and understanding of programming concepts.

II. Related Work

A. Tangible Robot Programming Tools for Programming Education

Algorithmic bricks (A-Bricks) is a tangible programming tool in the form of small blocks that can be connected horizontally or vertically [26]. Its purpose is to help students easily program the following concepts: sequence, repetition, condition and branch, function, and parameter. When 7-year-old students were tested using A-Bricks and Scratch, students who used A-Bricks had fewer errors and solved more problems than students who used Scratch. Moreover, students enjoyed using A-Bricks; usability scored more than 4, and satisfaction scored more than 4.5 (on a 1–5 scale).

Creative Hybrid Environment for Robotic Programming—CHERP, a tangible/graphic interface-based programming tool that can control LEGO blocks—was designed for children in kindergarten or elementary school [27]. Students can manipulate square blocks and compose a control flow that contains sequence, repetition, and condition and branch to solve robot-programming problems. When a test was done on 6-year-old students using an educational robot programming tool called TangibleK, their problem-solving ability and computational thinking improved [7].

The Programming Tangible Activity System—PROTEAS—kit, created at Aristotle University, Greece, is another tangible/graphic interface-based programming tool that can control LEGO robots [28]. Two types of tangible programming blocks, T butterfly and T ProRob, emulate programming concepts such as sequence, repetition, condition and branch, function, and variable when students use them. Students can combine tangible blocks to order robots to move forward, turn right or left, make sounds, or turn LED lights on or off. Furthermore, they can repeat sets, conditions, and nested loops. To measure the utility of this tangible programming tool, elementary school students were tested on solving robot problems using tangible and graphic programming tools. The results were then analyzed based on preference, enjoyment, usability, and error quantity. The students were separated into three groups based on their ages (5–6 years, 7–8 years, and 11–12 years), and all age groups showed a clear preference for tangible programming tools. However, the results showed that male students were more comfortable than female students with graphic programming tools. In addition, the older the student, the greater the preference for graphic programming tools [29]. According to the analysis of the robot problem solutions, the time taken to solve problems did not differ according to which tools were used, but using tangible programming tools resulted in fewer errors than using graphic programming tools.

Reviewing related research on elementary school students using tangible programming tools shows that tangible programming tools can be used in programming education and have a positive educational output. Nonetheless, it is important to continue researching ways to motivate students learning programming and to help them enjoy it.

B. Game-Based Learning for Programming Education

Edutainment merges education and entertainment concepts so that students engage in self-learning while playing, resulting in an increase in the quality of education [30]. When gaming is used in education, learners actively participate in lessons because education becomes entertaining. Eventually, this boosts educational output [31]. These benefits have led various fields, such as math and science, to use games for learning.

Similarly, programming education has various educational games for novice programmers. A game called Lightbot involves a combination of commands such as forward, turn, repeat, and function to move a robot to a desired location [8]. During the process of completing game tasks, students can simulate their programmed results and debug errors themselves. Thus, games are effective in improving computational thinking.

Another study was done on teaching methods by having students create their own games. For the purpose of learning object-oriented programming language, a course using a tangible game tool (Sifteo) was more efficacious than existing programming education [32]. When 12- and 13-year-old students were tested with the 3-D game-making tool Flip, their computational skills improved during the process of making games [33].

TABLE I
COMMAND BLOCKS IN A-BRICKS

Commands	Description
Forward	Robot moves 30cm forward
Turn Left	Robot turns 90 degrees left
Turn Right	Robot turns 90 degrees right
Sensor Forward	Robot moves forward until the sensor reaches the black line
Sensor Turn Left	Robot turns left until the sensor reaches the black line
Sensor Turn Right	Robot turns right until the sensor reaches the black line.
Function Defining	Defining the blocks connected on the right in functions.
Function Calling	Calling the functions registered.
Repetition Start	Setting the start location and the number of repetition
Repetition End	Setting the end location of repetition

The above research indicates that using games in education is effective in having students design and simulate algorithms to solve problems because the process combines student interest and enjoyment [12].

III. ROBOT GAME PROGRAMMING ENVIRONMENT

A. Components of the Proposed Programming Environment

1) Programming Tool Based on Tangible User Interface: A-Bricks, based on TUI, is a programming tool whose instructions even elementary school students understand intuitively. A-Bricks was developed for children to be able to experience programming activities through connecting and stacking command blocks [26]. Moreover, students can learn essential programming concepts (sequence, repetition, condition and branch, and function and parameter) spontaneously. Table I gives the ten A-Bricks commands and their descriptions.

2) Educational Robot: An educational robot should move accurately and sense conditions easily, according to a student's commands. Therefore, the robot uses two stepper motors to move forward or backward in a straight line, and to rotate 90° for turning left or right or moving at any angle that a student desires. In addition, infrared sensors are attached to the left and right of the front of the robot, so it can turn in the direction marked by students on a game board.

3) Game Board and Black Lines: The game board has eight types of 30 × 30 cm square plates. Each square plate has a distinct router pattern that, when combined with other plates, becomes a map for a robot mission. Therefore, student can create missions using these square plates and attempt various strategies to win the game.

The black line helps the robot to move forward or turn to the position and angle the user wants. During the mission, the black line and sensor blocks explained in Table I can be used to have the robot move forward freely along the problem-solving path and turn at various angles as required. Users must keep the robot's sensor location and angle in mind when placing the black line on the game board.

B. Robot Game Contents

1) Shortest Route Game: The shortest route game has students program the quickest route possible to reach designated spots. Its rules are as follows.

1) Teams or individuals take turns in placing plates to create the game board.
2) The game board edge is the starting point.
3) Players are only allowed to use the tangible programming blocks provided. (Levels can be adjusted by changing the number and the function of blocks within the provided set.)
4) A programmed robot can move twice on the same spot without changing the program.
5) If the robot departs from its course, it must return to the previous point and start again.
6) Only one robot at a time is allowed on the board.
7) If a robot stops moving, the other team's turn begins.
8) The winning team/individual's robot will reach all the destination spots in the shortest time.

2) Racing Game: In a racing game, students choose the start and finish points and make a path on a game board randomly constructed with game plates. The student who uses the least trials in programming wins. The rules are as follows.

1) Teams or individuals take turns in creating the game board.
2) On the game board created, the distance between the start and finish points is the farthest distance on the route.
3) Students pick five tangible programming tools and start programming.
4) If there is an error in the initial setup students have two chances to return to the start and reset.
5) If 50% of the robot's body leaves the route, it must start again from the beginning.
6) Only one robot at a time is allowed on the board.
7) All teams/individuals alternate turns until the game ends.
8) The team/individual's robot that reaches the finish quickest wins.

C. Robot Game Task for Learning Programming Concepts

Table II shows the programming concepts inculcated through the tasks executed in a robot game.

IV. EXPERIMENTS

A. Participants

The participants were 48 elementary school students. Eighteen were in the fourth grade (age = 10), 18 were fifth graders (age = 11), and 12 were sixth graders (age = 12).

B. Measurements

This paper adapted tools (usability, edutainment [23], [34], attitude [35], [36], and programming concepts [34], [37]) used in earlier research to be appropriate for an elementary student level. Three computer science education specialists were involved in the adjustment process. The final set of questions used to test usability, edutainment, and programming attitude

TABLE II
ROBOT GAME TASK FOR LEARNING PROGRAMMING CONCEPTS

Programming Concepts	Task in Robot Games
Sequence	Students must find the specific command bricks for the road section and connect them sequentially to construct the route for the game map.
Condition and Branch	Students find a branch in road section. And students use sensor bricks and attach black lines as a condition of determining the direction that the robot moves to the way students want.
Repetition	Students find the repeated sections of the route and use 'repetition start and end' bricks in order to reduce the number of bricks used in the game. (The game allows for using only the given number of command bricks.)
Function	Students find the same section of the constructed route and use a 'Function Defining' brick to make it into one consolidated brick to reduce the number of bricks used in the game.
Parameter	Students set a parameter of function bricks for number of repetitions in order to construct simultaneously repeat sections and same sections in the entire route.

are given in the Appendix. Their validity was evaluated by administering them in a pilot test to 20 fifth graders and 20 sixth graders.

1) Usability: The measurement of usability assessed satisfaction and memorization based on ISO-9241-171 for game environments. The satisfaction survey consisted of three questions, and the memorization survey consisted of two questions. A 5-point Likert-type scale was used for all questions, and the reliability of the survey was very high (Cronbach's $\alpha = 0.904$ for satisfaction and Cronbach's $\alpha = 0.905$ for cognitive memory).

2) Edutainment: The measurement of edutainment was a four-question survey on enjoyment of the programming activity, again using a 5-point Likert-type scale; Cronbach's α was 0.876.

3) Programming Attitude: The measurement of programming attitude was based on interest, confidence, and value for programming. Each factor included five questions for a total of 15 questions. A 5-point Likert-type scale was used on all questions, and survey reliability was high (Cronbach's $\alpha = 0.774$ in interest, Cronbach's $\alpha = 0.835$ in confidence, and Cronbach's $\alpha = 0.788$ in value).

4) Understanding of Programming Concepts: Understanding programming concepts was evaluated by testing five concepts: 1) sequence; 2) repetition; 3) condition and branch; 4) function; and 5) parameter. A total of ten problems were developed for the five concepts (two problems for each concept). The programming concept questions were presented in pseudo-code, in text and images appropriate for an elementary student level. Questions were paired in a two-tier test format [38], with the first type of questions requiring a multiple-choice or short-answer response, and the second asking the student's reason for giving the first response.

C. Procedure

A pretest measured students' programming attitudes and understanding of programming concepts. Then, students, in pairs, participated in 8 h of activities in an experimental class. Finally, a post-test measured usability, entertainment, programming attitudes, and understanding of programming concepts. The last of these, understanding of programming concepts, was only evaluated for the fifth- and sixth-grade students (30 students); it was not tested on fourth graders because they were not used to the two-tier test format questions and found it difficult to understand questions on the pseudo-code type of programming concepts.

TABLE III
USABILITY AND EDUTAINMENT QUESTIONNAIRE RESULTS

Factor		Questionnaire Item	Mean (SD)
Usability	Satisfaction	S-1	4.42(0.91)
		S-2	4.23(1.10)
		S-3	4.24(1.02)
	Cognitive Memory	M-1	4.52(0.71)
		M-2	4.58(0.73)
Edutainment		E-1	4.72(0.50)
		E-2	4.46(1.01)
		E-3	4.34(1.17)
		E-4	4.26(1.01)

N=48

V. RESULTS AND ANALYSIS

A. Usability and Edutainment in Tangible Robot Game Environment

Students' perspectives on the usability and edutainment aspects of a robot game environment are illustrated in Table III.

The results show that the usability and edutainment elements scored higher than 4, indicating that the proposed game environment is suitable for elementary-level students. More specifically, the result for cognition of memory was more than 4.5, which indicates that the usability of the proposed programming environment puts a minimal cognitive load on students. For the edutainment element of the environment, the highest result, 4.72, was for the question "I think it is fun to find the most effective method for winning the game" (E-1). This reveals that the proposed programming tool has a high educational efficiency for developing algorithmic thinking. However, the question "I think it is easy to solve the problems within the game environment" (s-2) received 4.23, the lowest score. Students found it relatively difficult to construct the path and to program to fulfill the mission. This result may reflect the complexity and difficulties of programming, but the score of 4 indicates that students are willing to overcome the difficulties of programming.

The results of the usability and edutainment elements of the proposed programming environment suggest that, by using entertaining games, elementary school students can usefully participate in programming activities and can achieve educational efficiency.

B. Programming Attitudes

Table IV illustrates the difference in students' attitudes about programming before and after participating in the robot game activity.

TABLE IV
STATISTICAL ANALYSIS OF PRE- AND POST-TEST PROGRAMMING ATTITUDES

Factor	test	25%ile	Mdn	75%ile	Z
Interest	Pre-	3.6	4.0	4.4	3.570***
	Post-	3.9	4.8	4.8	
Confidence	Pre-	3.4	3.8	4.4	3.669***
	Post-	3.7	4.6	5.0	
Value	Pre-	3.6	4.1	4.6	4.276***
	Post-	4.2	4.8	5.0	

N=48, Significant α=.05, *** p<.001

TABLE V
STATISTICAL ANALYSIS OF PRE- AND POST-UNDERSTANDING OF PROGRAMMING CONCEPTS

Factor		25%ile	Mdn	75%ile	Z
Sequence	Pre-	10.0	10.0	10.0	1.069
	Post-	10.0	10.0	10.0	
Repeat	Pre-	6.8	8.0	10.0	3.766***
	Post-	10.0	10.0	10.0	
Condition and Branch	Pre-	0.0	5.0	10.0	3.642***
	Post-	6.0	10.0	10.0	
Function	Pre-	0.0	0.5	1.0	1.679
	Post-	0.0	0.0	5.0	
Parameter	Pre-	1.0	6.0	10.0	3.570***
	Post-	10.0	10.0	10.0	
Total Score	Pre-	4.8	6.0	7.35	4.306***
	Post-	7.3	8.0	9.0	

N=30, Significant α=.05, *** p<.001

Students' post-test scores for interest, confidence, and value in programming attitudes significantly increased, with the medians all scoring above 4 above in post-test. The median of all programming attitudes factors increased by 0.7–0.8 points after the game activity, but the 25th percentile only increased by 0.3, except for the value factor. This indicates that students whose interest and confidence were low remained in that low position of the test group after the game activity; this shows that students' fundamental interest and confidence are important attributes to their programming attitude. However, for the value factor, the 25th percentile increased by 0.6 and reached a score of 4.2 in post-test; this result shows that tangible programming is more influential for students who did not appreciate the value of programming.

The proposed tangible programming environment thus has a positive influence on students' programming attitudes, and in particular helps them to perceive programming as a valuable and meaningful activity to partake in.

C. Understanding of Programming Concepts

Table V reports the understanding of programming concepts prior to and after the robot game activity, with the maximum score being a 10. Statistically, the understanding of programming concepts increased significantly after the robot game activity, in which students solved problems using repeat, condition and branch, and parameter concepts. These results show that students' self-directed learning of fundamental programming concepts can be achieved through participating in a robot game activity. Antithetically, sequence and function concepts can be analyzed to show no pre-/post-statistical difference. The concept of sequence can be easily understood at the elementary school student level, so the activity had little additional effect. Similarly, only a low-comprehension level is required for function, so no difference appears statistically. This lack of effect can be explained by students not having been previously exposed to the concepts of function and activity, and by the limited time students spent in the robot game environment not being enough for them to completely understand function concepts. In practice, to learn function concepts, students essentially have to experience making a function structure in problem-solving procedures. The experimental game activities did not provide these circumstances of using function blocks as a mandatory condition, so students were more focused on assembling the blocks sequentially for simple problem solving. Consequently, students were unsuccessful in learning function concepts. However, limiting students to using a certain number of blocks in the game activity, and requiring them to use function blocks by having them follow the game regulations strictly, enabled them to learn function concepts.

The use of a robot game environment for learning the fundamental programming concepts of sequence, repeat, condition and branch, and parameter resulted in a significant increase in programming education efficiency.

VI. CONCLUSION

This paper indicates that by providing three elements—effective educational tools such as an educational robot, games that can motivate and interest student learning, and a tangible interface suitable for elementary students' cognition—increases the prospects for an effective programming education for elementary students. Combining effective existing educational tools and learning methods with the proposed robot game environment achieved an effective educational synergy.

In programming education, providing game-format educational contents that motivate students' learning, such as those using robots, along with tools that simplify the programming process, can achieve the following effects. First, presentation of the concepts required in programming activities can be matched to the intellectual level of elementary school students to help them learn programming naturally. Second, interactive learning is exercised in this problem-solving process, and it can be noted that social scaffolding occurs in learning programming. Third, an interest in the problem-solving process based on computing can be encouraged, and students can come to understand that information technology is as critical for humanity as math and science, leading to a higher likelihood of students choosing a computer-related career. Thus, if elementary students were provided with the usability and edutainment of a robot game environment, it could decrease their cognitive load and increase their problem-solving abilities by helping them focus on the problem-solving activity.

In future research, it will be critical to develop robot game activities for programming learning for elementary students; it will be necessary to evaluate their behavior and results from a learning science perspective to achieve educational efficiency. Furthermore, it is critical to effectively combine all

efficient educational tools and teaching-learning methods of programming education to achieve synergy.

VII. Limitations

Limitations of this paper are that there were too few participants to generalize from, and that the results may be impacted by the research being conducted on Korean elementary students, who may vary from the norm in their experience with using information technology. The three areas of questioning (usability, edutainment, and attitude) were constructed positively based on a related question-based research design for elementary school students [39]. However, the use of positive type of questions may mean that an accurate measurement of students' perception is difficult. It is necessary to understand the results of the three areas in consideration of the type of question.

In order to evaluate the effectiveness of the proposed robot game environment for programming education within the experimental design, the assessment was done by asking questions about programming concepts. The results indicated that it was effective for learning branch, repetition, and parameter concepts, but not for understanding the function concept. However, in the actual class based on the experimental design, students were observed to use diverse problem-solving strategies to complete missions within the robot game programming activity. If assessments of problem-solving strategies were also considered in addition to the experimental design for the evaluation of programming education, the proposed robot game programming would have been clearly proven to have educational value.

Appendix

A. Programming Concepts Questions

The questions on the five concepts of programming (sequence, repetition, condition and branch, function, and parameter) are given below. The questions used were in Korean, and are translated here.

1) Sequence Concept:

1∼2. Jane wants to control the car to reach the end spot, using the computer.

	(1)		(8)	
(Start)	(2)	(9)	(10)	(11)
	(3)		(12)	
(6)	(4)	(7)	(13)	(End)
	(5)		(14)	

Available commands	
Go to the right	The car moves one space to the right.
Go to the left	The car moves one space to the left.
Go to the top	The car moves one space to the top.
Go to the bottom	The car moves one space to the bottom.

1-1. In order to control the car to the end spot, Jane constructed the following program.

```
Start of the car control program
   Control = Move to the right
   Control = Move to the bottom
   Control = [ⓐ]
   Control = Move to the right
   Control = [ⓑ]
   Control = Move to the right
End of the car control program
```

Write the appropriate control commands for ⓐ, ⓑ.

ⓐ : _____

ⓑ : _____

1-2. Why do you think this?

2-1. If Jane's car follows the program below, where would it be?

```
Start of the car control program
   Control= Move to the right
   Control= Move to the right
   Control = Move to the right
   Control= Move to the bottom
   Control = Move to the bottom
   Control = Move to the left
End of the car control program
```

Location number : #_____

2-2. Why do you think this?

2) Repeat Concept:

3∼4. Jane wants to control the car to reach the end spot, using the computer.

	(1)		(13)	
(Start)	(2)	(10)	(14)	(19)
	(3)		(15)	
(8)	(4)	(11)	(16)	(20)
	(5)		(17)	
(9)	(6)	(12)	(18)	(21)
	(7)		(End)	

Available commands	
Go to the right	The car moves one space to the right.
Go to the left	The car moves one space to the left.
Go to the top	The car moves one space to the top.
Go to the bottom	The car moves one space to the bottom.
Repetition (number of times) <Direction of Movement>	To move the car repeatedly (according to the number of times) in the programmed direction.

3-1. In order to control the car to the end spot, Jane constructed the following program.

```
Start of the car control program
   Control = Repeat[3] <Go to the right>
   Control = Repeat[ⓐ] <  ⓑ  >
End of the car control program
```

Write the appropriate control commands for ⓐ, ⓑ.

ⓐ : _____

ⓑ : _____

3-2. Why do you think this?

4-1. If Jane's car follows the program below, where would it be?

```
Start of the car control program
   Control = Move to the right
   Control = Repeat[2] <Move to the bottom>
   Control = Repeat[2] <Move to the right>
   Control = Move to the bottom
End of the car control program
```

Location number : #_____

4-2. Why do you think this?

3) Condition and Branch Concept:

5∼6. Jane has constructed a puzzle program using the computer. The puzzle program has the following rules.

<Rule>

1. The types of pieces used in the puzzle are tree, metal, and plastic.
2. Only the appropriate type of piece can be placed in the pre-planned location.

This is how the constructed puzzle program was executed.

<table>
<tr><td colspan="3">#1 (Tree)</td></tr>
<tr><td rowspan="2">#2 (Tree)</td><td>#4 (Metal)</td><td>#3 (Tree)</td></tr>
<tr><td colspan="2">$5(Metal)</td></tr>
<tr><td colspan="2">#7(Plastic)
#9(Plastic)</td><td>#8(Plastic)</td><td>#6(Metal)</td></tr>
</table>

5-1. Please complete the program that Jane planned.

```
Start the program
   Puzzle space = [1:9]
   Puzzle piece type = tree, metal, plastic
   If putting the puzzle together = [tree] example,
   Puzzle together = [ ⓐ ].
   If putting the puzzle together = [ ⓑ ] example,
   Puzzle together = #7, #8, #9.
   If putting the puzzle together = [ ⓒ ] example,
   Puzzle together = #4, #5, #6.
End of the program
```

ⓐ : _____

ⓑ : _____

ⓒ : _____

5-2. Why do you think this?

6. The program that Jane planned was adjusted by Tom in the following manner.

```
Start of the program
   Puzzle space = [1:9]
   Puzzle piece type = tree, metal, plastic
   If putting the puzzle together = [tree] example,
   Puzzle together = [ ⓐ ].
   If putting the puzzle together = [ ⓑ ] example,
   Puzzle together = #7, #8, #9.
   Puzzle together = [Rubber] example,
   Puzzle together = #4, #5, #6.
End of the program
```

6-1. Is it possible to operate the program as Tom adjusted it?

① Yes, it's possible.

② No, it's not possible.

6-1. Is it possible to operate the program as Tom adjusted it?

① Yes, it is possible.

② No, it is not possible.

6-2. What is the reason for your answer to 6-1?

① The computer was able to perceive the puzzle piece as rubber and operated automatically.

② Since rubber was not used previously as a puzzle piece, the program could not operate.

③ The computer was able to automatically switch the metal puzzle piece to rubber, so it operated.

④ The program operated successfully regardless of the type of puzzle piece used.

4) Function Concept:

7∼8. According to the image, the square includes a circle, triangle, star, and triangle.

7-1. If the sequence of the pieces is square, circle, triangle, square, and star:

How many  are there? _____

7-2. Why do you think this?

8-1. In between the star shapes, if there are three repetitions:

How many  are there? _____

8-2. Why do you think this?

5) Parameter Concept:

9∼10. Jane and Tom each have their own basket to hold the stars. Tom can hold two stars at a time and Jane can hold one star at a time.

9-1. If Tom and Jane were to place the stars three times, how many stars would be in the basket?

How many stars are in Tom's basket _____?

How many stars are in Jane's basket _____?

or make

9-2. Why do you think this?

10-1. If there are ten stars in the Tom's basket, how many stars would be in Jane's?

How many stars are in Jane's basket _____?

10-2. Why do you think this?

B. Questionnaire

1) Usability Questionnaire:

(S-1) I think that using the game environment is convenient.

(S-2) I think it is easy to solve the problems within the game environment.

(S-3) I think it is easy to construct and control the game environment.

(M-1) I can easily explain to others how to use the game environment.

(M-2) I think the game environment instructions are easy to remember.

2) Edutainment Questionnaire:

(E-1) I think that it is fun to find the most effective method for winning the game.
(E-2) I think the game environment is amusing.
(E-3) I want to continue using the game environment.
(E-4) I want to play this game with other students who have not done so.

3) Programming Attitude Questionnaire:

(V-1) I think that I can resolve practical problems through robots and programming.
(V-2) I think that I can live a successful life if I understand robots and programming.
(V-3) I think that robots and programming are important for constructing something new in the future.
(V-4) I think that robots and programming will have a critical role in my life.
(V-5) I think that today's experiential learning is as important as learning math and science.
(I-1) I want to ask questions of, or learn from, a person who is proficient at robots and programming whenever I see him or her.
(I-2) I want to befriend someone who is proficient at robots and programming.
(I-3) I want to obtain a job that uses robotic and programming skills.
(I-4) I enjoy using the concepts learned from robot and programming activities.
(I-5) I want to continue learning robots and programming.
(C-1) I am confident of understanding contents related to robot and programming.
(C-2) I am confident that I can be proficient at learning robots and programming.
(C-3) I am confident that I can solve problems and tasks using robots and programming.
(C-4) I am confident that I can be proficient at doing something using robot and programming.
(C-5) I am confident that I can simplify a problem, or make it less difficult, using robots and programming.

REFERENCES

[1] S. Y. Lye and J. H. L. Koh, "Review on teaching and learning of computational thinking through programming: What is next for K-12?" *Comput. Human Behav.*, vol. 41, pp. 51–61, Dec. 2014.

[2] D. Barr, J. Harrison, and L. Conery, "Computational thinking: A digital age skill for everyone," in *Proc. Int. Soc. Technol. Educ. (ISTE)*, 2011. [Online]. Available: http://www.iste.org/docs/learning-and-leading-docs/march-2011-computational-thinking-ll386.pdf

[3] J. M. Wing, "Computational thinking and thinking about computing," *Philosoph. Trans. Roy. Soc. A Math. Phys. Eng. Sci.*, vol. 366, pp. 3717–3725, Jul. 2008.

[4] R. Wachenchauzer, "Work in progress—Promoting critical thinking while learning programming language concepts and paradigms," in *Proc. IEEE Conf. Publ. Front. Educ.*, Savannah, GA, USA, 2004, pp. 13–14.

[5] W.-Y. Hwang, R. Shadiev, C.-Y. Wang, and Z.-H. Huang, "A pilot study of cooperative programming learning behavior and its relationship with students' learning performance," *Comput. Educ.*, vol. 58, no. 4, pp. 1267–1281, 2012.

[6] M. A. Brito and F. de Sá-Soares, "Assessment frequency in introductory computer programming disciplines," *Comput. Human Behav.*, vol. 30, pp. 623–628, Jan. 2014.

[7] M. U. Bers, L. Flannery, E. R. Kazakoff, and A. Sullivan, "Computational thinking and tinkering: Exploration of an early childhood robotics curriculum," *Comput. Educ.*, vol. 72, pp. 145–157, Mar. 2014.

[8] C. Kazimoglu, M. Kiernan, L. Bacon, and L. MacKinnon, "Learning programming at the computational thinking level via digital game-play," *Procedia Comput. Sci.*, vol. 9, pp. 522–531, Dec. 2012.

[9] J. Tan, X. Guo, W. Zheg, and M. Zhong, "Case-based teaching using the laboratory animal system for learning C/C++ programming," *Comput. Educ.*, vol. 77, pp. 39–49, Aug. 2014.

[10] J. Moons and C. D. Backer, "The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism," *Comput. Educ.*, vol. 60, no. 1, pp. 368–384, 2013.

[11] S.-S. Abdul-Rahman and B. du Boulay, "Learning programming via worked-examples: Relation of learning styles to cognitive load," *Comput. Human Behav.*, vol. 30, pp. 286–298, Jan. 2014.

[12] J. Denner, L. Werner, and E. Ortiz, "Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts?" *Comput. Educ.*, vol. 58, no. 1, pp. 240–249, 2012.

[13] T. S. McNerney, "From turtles to tangible programming bricks: Explorations in physical language design," *Pers. Ubiquitous Comput.*, vol. 8, no. 5, pp. 326–337, 2004.

[14] M. Resnick *et al.*, "Scratch: Programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, 2009.

[15] E. Hornecker and J. Buur, "Getting a grip on tangible interaction: A framework on physical space and social interaction," in *Proc. ACM CHI*, Montreal, QC, Canada, 2006, pp. 437–446.

[16] H. Ishii and B. Ullmer, "Tangible bits: Towards seamless interfaces between people, bits and atoms," in *Proc. Conf. Human Factors Comput. Syst.*, Atlanta, GA, USA, 1997, pp. 234–241.

[17] P. Marshall, "Do tangible interfaces enhance learning?" in *Proc. Int. Conf. Tangible Embedded Interact.*, Baton Rouge, LA, USA, 2007, pp. 163–170.

[18] O. Zuckerman, S. Arida, and M. Resnick, "Extending tangible interfaces for education: Digital montessori-inspired manipulatives," in *Proc. SIGCHI*, Portland, OR, USA, 2005, pp. 859–868.

[19] Y. Fernaeus and J. Tholander, "Looking at the computer but doing it on land: Children's interactions in a tangible programming space," in *Proc. HCI*, Las Vegas, NV, USA, 2005, pp. 3–18.

[20] S. Atmatzidou, I. Markelis, and S. Demetriadis, "The use of LEGO mindstorms in elementary and secondary education: Game as a way of triggering learning," in *Proc. Int. Conf. Simulat. Model. Program. Auton. Robots*, Venice, Italy, 2008, pp. 22–30.

[21] L. Huang, T. Varnado, and D. Gillan, "Practices of teaching problem solving skills in robotics education," *Proc. Human Factors Ergonom. Soc. Annu. Meeting*, vol. 57, no. 1, pp. 1830–1834, 2013.

[22] R. Garris, R. Ahlers, and J. E. Driskell, "Games, motivation, and learning: A research and practice model," *Simulat. Gaming*, vol. 33, no. 4, pp. 441–467, 2002.

[23] C. Pelletier, "Games and learning: What's the connection?" *Int. J. Learn. Media*, vol. 1, no. 1, pp. 83–101, 2009.

[24] R. E. Mayer and M. C. Wittrock, "Problem solving," in *Handbook of Educational Psychology*, P. A. Alexander, P. H. Winne, and G. D. Phye, Eds. Mahwah, NJ, USA: Erlbaum, 2006, pp. 287–303.

[25] D. Xu, D. Blank, and D. Kumar, "Games, robots, and robot games: Complementary contexts for introductory computing education," in *Proc. GDCSE*, Miami, FL, USA, 2008, pp. 66–70.

[26] D.-Y. Kwon, H.-S. Kim, J.-K. Shim, and W.-G. Lee, "Algorithmic bricks: A tangible robot programming tool for elementary school students," *IEEE Trans. Educ.*, vol. 55, no. 4, pp. 474–479, Nov. 2012.

[27] M. Bers and M. Horn, *Tangible Programming in Early Childhood: Revisiting Developmental Assumptions Through New Technologies*. Greenwich, CT, USA: Inf. Age, 2010.

[28] T. Sapounidis and S. Demetriadis, "Tangible versus graphical user interfaces for robot programming: Exploring cross-age children's preferences," *Pers. Ubiquitous Comput.*, vol. 17, no. 8, pp. 1775–1786, 2013.

[29] T. Sapounidis, S. Demetriadis, and I. Stamelos, "Evaluating children performance with graphical and tangible robot programming tools," *Pers. Ubiquitous Comput.*, vol. 19, no. 1, pp. 225–237, 2015.

[30] J. Bourgonjon *et al.*, "Acceptance of game-based learning by secondary school teachers," *Comput. Educ.*, vol. 67, no. 1, pp. 21–35, Sep. 2013.

[31] F. Bertacchini, E. Bilotta, P. Pantano, and A. Tavernise, "Motivating the learning of science topics in secondary school: A constructivist edutainment setting for studying chaos," *Comput. Educ.*, vol. 59, no. 4, pp. 1377–1386, 2012.

[32] J. M. R. Corral, A. C. Balcells, A. M. Estévez, G. J. Moreno, and M. J. F. Ramos, "A game-based approach to the teaching of object-oriented programming languages," *Comput. Educ.*, vol. 73, no. 1, pp. 83–92, 2014.
[33] K. Howland and J. Good, "Learning to communicate computationally with flip: A bi-modal programming language for game creation," *Comput. Educ.*, vol. 80, pp. 224–240, Jan. 2015.
[34] J.-M. Sáez-López, M. Román-González, and E. Vázquez-Cano, "Visual programming languages integrated across the curriculum in elementary school: A two year case study using 'Scratch' in five schools," *Comput. Educ.*, vol. 97, pp. 129–141, Jun. 2016.
[35] S. Wiedenbeck, "Factors affecting the success of non-majors in learning to program," in *Proc. 1st Int. Workshop Comput. Educ. Res.*, Seattle, WA, USA, 2005, pp. 13–24.
[36] A. E. Tew, B. Dorn, and O. S. Schneider, "Toward a validated computing attitudes survey," in *Proc. 9th Annu. Int. Conf. Int. Comput. Educ. Res.*, Auckland, New Zealand, 2012, pp. 135–142.
[37] F. Kalelioğlu, "A new way of teaching programming skills to K-12 students: Code.org," *Comput. Human Behav.*, vol. 52, pp. 200–210, Nov. 2015.
[38] T.-C. Yang, S. Y. Chen, and G.-J. Hwang, "The influences of a two-tier test strategy on student learning: A lag sequential analysis approach," *Comput. Educ.*, vol. 82, pp. 366–377, Mar. 2015.
[39] H. W. Marsh, "Negative item bias in ratings scales for preadolescent children: A cognitive-developmental phenomenon," *Develop. Psychol.*, vol. 22, no. 1, pp. 37–49, 1986.

Jaekwoun Shim received the B.Ed. degree from the Gyeongin National University of Education, Incheon, South Korea, in 2007, and the M.S. degree in computer science education, Korea University, Seoul, South Korea, in 2012, where he is currently pursuing the Ph.D. degree.

His current research interests include computational thinking and programming education.

Daiyoung Kwon received the B.S., M.S., and Ph.D. degrees in computer science education, Korea University, Seoul, South Korea, in 2000, 2006, and 2011, respectively.

He is a Research Professor with the Department of Computer Science Education, College of Education, Korea University, where he was a Research Professor with the Creative Informatics and Computing Institute from 2011 to 2013. His current research interests include algorithmic thinking learning, educational programming languages, and cognitive experiments for evaluating thinking abilities.

Wongyu Lee received the B.A. degree in English language and literature from Korea University, Seoul, South Korea, and the M.Eng. and Dr.Eng. degrees in information engineering from the University of Tsukuba, Tsukuba, Japan.

He is a Professor with the Department of Computer Science and Engineering, College of Informatics, Korea University. He was with Culture Informationization in the Korean Culture and Arts Foundation, Seoul. His current research interests include informatics education, representation of structuralized information, information management, and education policy.