The Effect of Programming Classes with Tangible Scratch Blocks on the Programming Interest of 6th Grade Elementary School Students

Seok-Ju Chun, Yunju Jo, and Seungmee Lee

***Abstract*—In this paper, we introduce an original, classroom-based approach for teaching Scratch programming to 6th grade elementary school students. Scratch is a programming language that involves assembling icon-based command blocks. It was designed to avoid the complex syntax errors seen in other programming languages, making it especially accessible for younger learners. While Scratch does provide a visual programming environment in which potentially just about anyone can learn to read and write programming code, there can still be a reduced overall interest in learning programming, because younger learners in particular can find it difficult to intuitively understand or be stimulated by abstract concepts of programming such as sequences, conditions, and repetition, which are present in Scratch. Our research involves the development of a tangible, electronic block system that allows students to manipulate physical objects with their hands to perform programming tasks. The system consists of a Scratch simulator and physical, Scratch electronic blocks embodying Scratch user interface shapes. We devised and delivered a programming course to 6th grade Korean elementary school students using our block system. The results are encouraging.**

***Index Terms*—Scratch programming, tangible block programming, electronic block system, programming education, elementary school students' programming class.**

I. Introduction

With the progression of the 4th industrial revolution, the importance of computing technology such as AI, big data, and cloud computing continues to grow. Computer science education is critical for nurturing the next generation of tech experts, and various studies have been conducted to better promote computer science education in schools [1], [2].

In the case of elementary schools, block-based programming languages are frequently employed to develop students' computational thinking [3]. Among these, Scratch [4] is very popular and available in more than 40 languages and 150 countries. Scratch is based on a Graphical User Interface (GUI), and it is highly suited to complete newcomers to computer programming because it minimizes grammatical errors, thus is potentially simpler to learn [5], [6].

However, younger learners in particular can still feel a cognitive burden when it comes to how to interact with the Scratch interface and programming concepts, because these can appear very advanced and abstract (e.g., the concepts of sequence, loops, and conditionals) [7]. According to an analysis of Scratch use among 4th–6th grade students, students can take a long time to find blocks to use in Scratch interface. Also, the longer the codes are, the harder it is for students to understand the relationship between blocks [8], [9]. In addition, one study found that students' perceptions of programming change for the worse after block-based programming classes, and their overall motivation and enjoyment decrease [10].

To address these problems, we developed physical Scratch blocks that allow students to program by assembling Scratch blocks directly with their hands. This is based on a Tangible User Interface (TUI) concept, a concept which allows computer system users to interact with digital content through the manipulation of tangible objects [11]. Using our blocks, we taught classes to 6th grade elementary school students. We evaluated the students' interest in programming through a survey before and after classes, and we interviewed the students at the end of the course. Our hope is that elementary school students will learn Scratch programming more easily and more enjoyably by taking advantage of our system.

II. Related Work

Scratch [4] is an educational programming language developed by the Lifelong Kindergarten Group at MIT Media Lab in the U.S. It is based on a GUI, meaning users make algorithms by clicking on or dragging and dropping blocks on a computer screen. Scratch is appropriate for novice programmers to learn the basic principles of programming (sequences, conditionals, and loops) because it presents fewer grammatical and logical errors than other programming languages. It provides an effective way for elementary school students to learn coding and programming because of being more accessible and the appeal of creating various multimedia projects [6].

For beginners, however, there is still a relatively high cognitive burden when it comes to the Scratch interface. Scratch presents a number of different kinds of blocks needed for programming on the screen, so it takes students a long time to find the blocks they need [8]. Also, because command blocks are presented graphically, the longer and more complex the connection of the command blocks, the more

Manuscript received November 25, 2020; revised June 12, 2021.

Seok-Ju Chun is with Seoul National University of Education, Republic of Korea (e-mail: chunsj@snue.ac.kr).

Yunju Jo is with Sin-Mook Elementary School, Republic of Korea (e-mail: yunju0514@gmail.com).

Seungmee Lee is with Guui Elementary School, Republic of Korea (e-mail: me1226@sen.go.kr).

difficult it is for students to understand the relationship between the blocks [9]. To aid with overcoming concepts that are difficult and abstract, intuitive manipulation becomes effective [12]. Therefore, in this work, Scratch 3.0 blocks were implemented as physical electronic blocks. Users can produce code by touching and connecting these physical electronic blocks with their hands, and the results are immediately verified by the simulator.

These physical electronic blocks are tangible programming tools. A TUI-based learning environment is an environment that helps coders understand difficult concepts by lowering abstract concepts to a level that can be easily manipulated in a physical environment using the body [12].

Various tangible programming tools have been developed to help students understand abstract programming concepts (sequences, conditionals, loops, variables, functions, etc.) through specific manipulation activities, and their effectiveness has been published in various work [13]-[16].

Tern [15] is a tangible programming tool for writing code by combining pieces of puzzle-shaped commands. Comparing the task performance of students and adult participants in programming classes using Tern and Scratch, students using Tern solved problems better than students using Scratch. According to student interviews, students reported that touching and manipulating wooden puzzles felt like a fun game, and that touching a real puzzle was more enjoyable than manipulating a mouse.

Toque [16] is a cooking-based programming language that uses the Nintendo Wiimote and Nunchuk. Users can open and close a Loop via the Wiimote's *up* and *down* buttons and control the number of counts in a Loop via the + and - buttons. The programming results can be viewed on the screen. Toque provides a good environment for learning a procedural workflow, but it does not have enough learning content.

TurTan [14], based on Logo, is a tangible programming system designed for turtle geometry. TurTan is designed to make it easier for 4–7 year olds to understand the basic principles of programming and to enjoy learning programming. However, even though it is intended for children, the tool use is complex, and it is expensive to purchase an interactive desktop. Therefore we developed Scratch electronic blocks as part of a tangible programming toolkit targeted at elementary school students.

III. The Scratch Electronic Block System

Our Scratch electronic block system consists of one event block and several kinds of command blocks. We designed our electronic blocks to mimic the Scratch blocks provided by MIT Scratch 3.0 (Fig. 1) in terms of their shape and functionality. Our Scratch electronic block solution allows users to connect blocks with their hands just like LEGO blocks instead of dragging and dropping virtual blocks in a GUI-based Scratch programming environment using a mouse. The blocks are magnetic and connect to each other easily. They are similar to their virtual counterparts in functionality. After connecting an event block to several command blocks, a user can push a green flag button and trigger the event block to communicate with the command blocks and read the overall block structure.

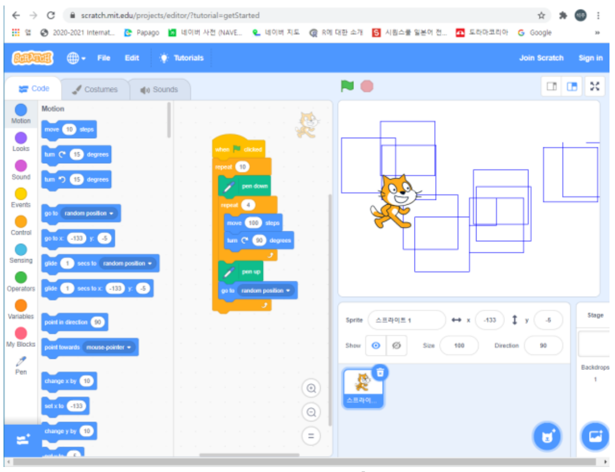
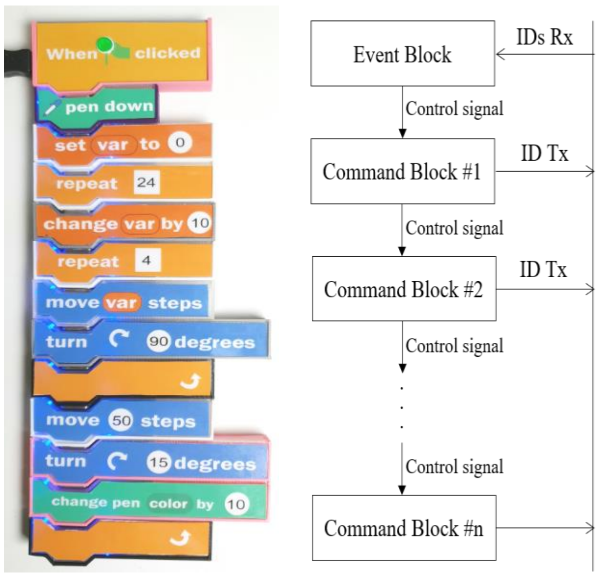
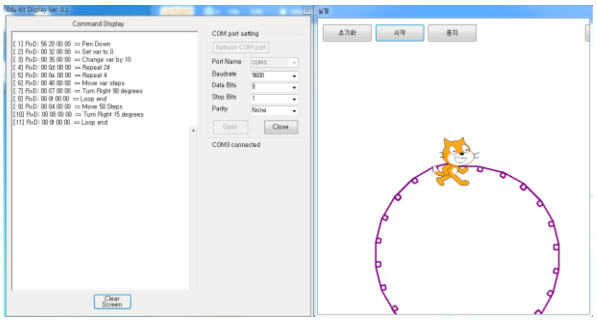


Fig. 1. MIT Scratch 3.0.



(a) Scratch Electronic Blocks (b) System Structure

Fig. 2. Scratch electronic blocks.



```
[ 1] RxD: 56 20 00 00  == Pen Down
[ 2] RxD: 00 32 00 00  == Set var to 0
[ 3] RxD: 00 35 00 00  == Change var by 10
[ 4] RxD: 00 0d 00 00  == Repeat 24
[ 5] RxD: 00 0a 00 00  == Repeat 4
[ 6] RxD: 00 40 00 00  == Move var steps
[ 7] RxD: 00 07 00 00  == Turn Right 90 degrees
[ 8] RxD: 00 0f 00 00  == Loop end
[ 9] RxD: 00 04 00 00  == Move 50 Steps
[10] RxD: 00 08 00 00  == Turn Right 15 degrees
[11] RxD: 00 0f 00 00  == Loop end
```

Port Name: COM3
Baudrate: 9600
Data Bits: 8
Stop Bits: 1
Parity: None

COM3 connected

Fig. 3. Scratch simulator.

Fig. 2 shows the connected Scratch electronic blocks and the system structure. When a user completes programming with Scratch electronic blocks in a tensible manner, the Scratch electronic block system starts operation by the pressing of the green button on the event block. Initially, the event block sends a control signal to identify the ID (identification) of the command block directly below it. When the command block receives a control signal from the event block, it sends its own ID to the event block and sends the control signal to the command block connected below it. In this same way, when the bottom-most command block receives a control signal, it sends its ID to the event block.

Each time an event block receives an ID from the command block, it is sent to the Scratch simulator.

As shown in Fig. 3, when the scratch simulator obtains ID information of all command blocks from the event block, it interprets the sequence of all the IDs (that is, the algorithm) so that sprites (images) move around.

Our work adopted 22 electronic blocks for teaching 6th graders Scratch programming in an elementary school classroom in Korea. To do this, we first analyzed the CS Framework's K-12 standard [17]. We chose the Algorithm and Programming Concept as our core for the lessons out of the five concepts available in the CS Framework [17]. From our chosen concept, we then selected the associated goal in the grades 6–8 (ages 11–14) band: "Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals." Considering the 6th grade level in elementary education, we finalized the choice of Scratch 3.0 blocks for teaching sequences, loops, and conditionals. Table I lists the details of the Scratch electronic blocks we used.

TABLE I: SCRATCH ELECTRONIC BLOCKS FOR 6TH GRADE STUDENTS

Category		Implemented electronic blocks
Event block		When flag clicked
Command blocks	Control	Forever Repeat 4 Repeat 24 If-then If-then-else Wait 1 sec Wait 2 sec
	Motion	Move 50 steps Move 100 steps Turn right 15 degrees Turn right 90 degrees Go to random position
	Sound	Play sound *meow* Play sound *record*
	Sensing	Touch mouse-pointer
	Variable	Set var to 0 Set var to 1 Change var by 1 Change var by 10
	Pen	Pen down Pen up

Our Scratch electronic blocks utilize most of the existing Scratch 3.0 blocks, although some differences exist around the shape of the blocks. Where a Scratch block represents a pair of commands (e.g., looping commands like *forever* and *repeat* or conditional commands like *if-then* and *if-then-else*), we implemented them as separate blocks. The differences are minor and do not cause any issues when students use the blocks to undertake Scratch programming.

IV. EXPERIMENT

Our research subjects comprised sixteen South Korean 6th grade elementary students (i.e., 12 year olds). We all gathered in a classroom to work through our designed course for three weeks on Fridays for 80 minutes per meeting in June and July 2020. All students who participated in our course had prior experience in block coding, and one student had experience in using Scratch. In the first class, all students received a pre-test that measured their interest of programing, and in the last class, they received a post-test in the form of asking about their interest of programing in addition to conducting an interview.

Our research question was: "Will the use of tangible Scratch electronic blocks in a taught course affect students' interest in programming?" Our hypothesis was: "Students will have a greater interest in programming after the course than before the course."

TABLE II: SCRATCH PROGRAMMING COURSE FOR 6TH GRADE STUDENTS

Session	Syllabus
1	·Course overview ·Pre-test: interest in programming
2	·Learning the basic concept of programming "Sequence" ·Move 50, Move 100, Move 150, Move 200 ·Draw a line by 50, change the color and draw a new line by 100 ·Arrange blocks appropriately to make cat move 50 to the right, turn 90˚, and play "Hello" sound
3	·Learning the basic concept of programming "Loop & Nested Loop" · Draw rectangles using "repeat 4 times" ·Draw 4 different rectangles · Draw figures using "go to random position" block ·Draw 10 squares at random locations, and make code as short as possible
4	· Learning the basic concept of programming "Events" · Understanding how to use the "If–then" and "If-then-else" block ·Let cat move 50 if touching mouse-pointer ·Arrange blocks appropriately to make cat say *meow* when it touches the mouse pointer while moving 200
5	·Draw own picture using tangible Scratch electronic blocks ·Verify result and share with peers
6	·Post-test: interest in programming ·Interview

To explore these questions, we designed course content to meet the CS Framework standard, and we used our tangible Scratch electronic blocks. The course comprises a total of six sessions, and in them, students learn about sequences, loops, and conditionals. Table II shows how our syllabus develops over the sessions. The course consists of activities that offer experience in implementing simple programs with the tangible Scratch electronic blocks and simulators, and correcting errors in already-made programs. In the fifth class, students worked on a personal art project. Students assembled their tangible blocks and displayed their results in a simulator.

As shown in Fig. 4, we shared a video of assembling blocks to implement algorithms with the students.

We assessed the level of interest in programming before and after the course by issuing a survey. In it, students expressed their level of interest by responding to survey questions, marking responses on a five-point Likert scale. The survey consisted of a total of nine questions and was developed by ourselves. The survey questions were oriented to the areas of: interest toward programming, interest toward programming education, interest toward programming activities, willingness to continue programming class, interest toward programming-related careers, and anxiety about programming lessons.

Fig. 4. Demonstration of scratch electronic blocks in a 6th grade classroom.

TABLE III: THE 6TH GRADE PRE-POST TEST RESULT OF INTEREST IN PROGRAMMING

	Mean	SD	T	P
Pre-class	3.4	.947	-3.393	.000**
Post-class	3.769	.208		

To prove the effectiveness of our course, we assessed the changes before and after the course We conducted a paired t-test for a single group. Table III shows the result. The *p*-value is 0.000 and less than 0.05. Therefore, there is a significant statistical difference between before and after. The students' interest in programming improved from 3.4 to 3.769 out of a five-point scale. Students were also more attentive during their classes because they were eager to test their results using the simulator after finishing the tangible block-based coding.

Students were also interviewed about their reflections on their use of the tangible Scratch electronic blocks, and this also revealed a heightened curiosity and interest in Scratch programming. Below are some extracts from the student interviews.

"It was amazing to program in a different way than usual. And if I had the chance, I would like to make my own game with these tangible blocks."

"It was great to program using three-dimensional things, and it felt like it was a game."

"I want to do programming using tangible blocks at home. Even if there was an error in the programming, I could correct it quickly with my hands."

This curiosity and interest in turn led to an increased concentration in the programming classes.

Programming based on a TUI is more effective for getting students to associate programming as playing or a game than programming based on a GUI, which refers to manipulating drag-and-drop command blocks on a computer screen.

V. CONCLUSION

In our study, Scratch electronic blocks that made use of tangible programming language were used to enhance programming learning. Elementary school students were able to experience Scratch programming immersively by assembling physical Scratch electronic blocks in their hands. A programming course using the tangible Scratch electronic blocks was developed and delivered and the students' programming interest was analyzed. The analysis showed that the students' interest level increased from 3.4 to 3.769, and the concentration during class was likewise greater.

Following end-of-course interviews, the 6th grade students with prior experience in block coding based on a GUI felt that programming felt more like a fun game when using the tangible Scratch blocks based on a TUI.

This study only involved a total of 16 students in a 6th grade class, so it is difficult to generalize and fully interpret the effectiveness of the tangible Scratch electronic blocks. Therefore, it is important to apply our programming classes using tangible Scratch electronic blocks to more students and to analyze the effectiveness. A comparative analysis study of differences in programming interest by dividing the same course content into GUIs and TUIs would also be useful.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Dr. Chun managed the project and designed the programming courses. Ms. Jo supervised the programming class and analyzed the results. Ms. Lee taught the programming courses and administered students survey. All authors co-wrote the paper and approved the final version.

ACKNOWLEDGMENT

This work was supported by the 2021 Research Fund of Seoul National University of Education.

REFERENCES

[1] Swkim and Yjlee, "Deveolment and application of arduino-based education program for high school students," *Journal of Theoretical & Applied Information Technology*, vol. 95, no. 18, 2017.
[2] Swkim and Yjlee, "The effect of robot programming education on attitudes towards robots," *Indian Journal of Science and Technology*, vol. 9, no. 24, pp. 1-11, 2016.
[3] N. Smith, C. Sutcliffe, and L. Sandvik, "Code club: Bringing programming to UK primary schools through scratch," *SIGCSE*, pp. 517-522, 2014.
[4] Scratch. [Online]. Available: http://scratch.mit.edu.
[5] M. Rednick, J. Maloney, A. M. Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, and J. Silver. "Scratch: Programming for all," *Communication of the ACM*, vol. 52, no. 11, pp. 60-67, 2009.
[6] N. B. Dohn, "Students' interest in Scratch coding in lower secondary mathematics," *Br. J. Educ. Technol*, vol. 51, no. 1, pp. 1-83, 2020.
[7] J. Moons and C. 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.
[8] C. Hill, H. A. Dwyer, T. Martinez, D. Harlow, and D. Franklin, "Floors and flexibility: Designing a programming environment for 4th-6th grade classrooms," *SIGCSE*, pp. 546-551, 2015.
[9] D. Franklin, J. Salac, C. Thomas, Z. Sekou, and S. Krause, "Eliciting student scratch script understandings via scratch charades," *SIGCSE*, pp. 780-786, 2020.
[10] J. Denner, L. 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.
[11] M. Paul, R. Yvonne, and H. Eva, "Are tangible interfaces really any better than other kinds of interfaces?" *In CHI'07 Workshop on Tangible User Interfaces in Context & Theory*, 2007.
[12] M. U. Bers, L. Flannery, E. R. Kazakoff, and A. Sullivan, "Computational thinking and tinkering: Exploration of an early

childhood robotics curriculum," *Computer & Education*, vol. 72, pp. 145-157, 2014.

[13] M. Bers and M. Horn, "Tangible programming in early childhood: Revisiting developmental assumptions through new technologies," *Greenwich*, CT: Information Age Publishing, 2010.

[14] D. Gallardo *et al*., *TurTan: A Tangible ´Programming Language for Creative Exploration*, IEEE Computer Society Press, pp. 89–92, 2008.

[15] S. H. Michael, *Comparing the Use of Tangible and Graphical Programming Languages for Informal Science Education*, ACM CHI, pp. 975-984, 2009.

[16] S. Tarkan, V. Sazawal, A. Druin *et al*., "Toque: Designing a cooking-based programming language for and with children," *CHI*, ACM, pp. 2417–2426, 2010.

[17] The K-12 Computer Science Framework. [Online]. Available: http://k12cs.org

Seok-Ju Chun is received the PhD degree in information and communication engineering from KAIST. From 1997 to 2004, he was an assistant professor in the Department of Internet Information at the Ansan University, Korea. Since 2004, he has been a professor of computer education at Seoul National University of Education (SNUE), Korea. Dr. Chun is the PI of the "Development and Application of Coding Robot Kit for Silver Care in Aging Society" project supported by the NRF. His current research interests include programming education, AI education, OLAP, and data mining.

Yunju Jo is Sin-Mook elementary school teacher. From 2014 to 2020, She worked as an elementary teacher in Seoul Metropolitan Office of Education(SMOE). She is received the master`s degree in computer education from Seoul National University of Education (SNUE) in 2017. She is taking a Ph.D in computer education at SNUE. She is the master of SW education at the SMOE. Her current research interests include programming education, AI education, and online learning.

Seungmee Lee is Guui elementary school teacher. From 2016, She worked as an elementary teacher in Seoul Metropolitan Office of Education(SMOE). She is received the bachelor`s degree in computer education from Korea National University of Education (KNUE) in 2016. She is taking a master`s degree in computer education at Seoul National University of Education (SNUE). Her current research interests include programming education, AI education.