Analyzing Viscuit Programs Crafted by Kindergarten Children

Takeshi Watanabe
watanabe@viscuit.com
University of Electro-Communications
Digitalpocket LLC.
Tokyo, Japan

Yuriko Nakayama
kagawafujimi@kdp.biglobe.ne.jp
Kagawa-fujimigaoka Kindergarten
Kanagawa, Japan

Yasunori Harada
hakase@viscuit.com
Digitalpocket LLC.
Kanagawa, Japan

Yasushi Kuno
y-kuno@uec.ac.jp
University of Electro-Communications
Tokyo, Japan

ABSTRACT

Viscuit is a programming language developed in Japan. With Viscuit, children can move and change multiple pictures on the screen using picture rewriting rules. Viscuit's distinguishing feature is that the program is expressed solely with pictures and their layouts without any letters. Therefore, Viscuit is very suitable for pre-school children. As an experiment, we carried out regular programming lessons using Viscuit in a kindergarten for one year in 2017. Through these lessons, we taught four Viscuit techniques. In this report, we analyze the programs made by 5-6 year old children in the final lesson of the 13 lessons. Our research question is how kindergarten children make Viscuit programs to express their ideas. We analyzed the kinds of programs children made to express their ideas by applying what they have learned in the previous 12 lessons. We let children make programs based on the operetta in which they were going to perform in the kindergarten graduation ceremony. As a result, 85.7% of children made valid programs and used multiple programming techniques effectively.

CCS CONCEPTS

• **Social and professional topics → K-12 education**; *Computational thinking*; • **Software and its engineering →** *Visual languages*.

KEYWORDS

viscuit, kindergarten, programming education, visual language, computational thinking

ACM Reference Format:
Takeshi Watanabe, Yuriko Nakayama, Yasunori Harada, and Yasushi Kuno. 2020. Analyzing Viscuit Programs Crafted by Kindergarten Children. In *Proceedings of the 2020 International Computing Education Research Conference (ICER '20), August 10–12, 2020, Virtual Event, New Zealand.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3372782.3406253

1 INTRODUCTION

Nowadays, we recognize the importance of programming education for young children [19, 21, 27, 28], and we needed to reveal to what extent they can understand and make programs [2, 5, 30]. Many studies on programming education for preschoolers report that children can make programs and are motivated enough when they perform programming [6, 22, 23]. However, there are few studies analyzing quantitatively what kind of programs children make and whether children understand what they are doing [5, 10, 36].

We designed a series of curricula to teach programming using Viscuit [12, 33], which adopts the picture rewriting rules to make programs. Through the lessons, we taught four Viscuit techniques. We taught this series of lessons to kindergarten children for one year. The participants were 28 5-6 year old children.

Our research question is how the children who learned programming in Viscuit express their ideas in the final lesson of the series of 13 lessons. In the final lesson, children created programs that introduce the operetta for their parents. They were going to perform this operetta at the kindergarten graduation ceremony. We expected the children to use effectively the programming techniques that they were taught. We analyzed how they applied the techniques to their programs.

Our analysis confirmed that most children were able to create programs and effectively express the story of the operetta by creating at least one successful work.

Section 2 introduces related work. Section 3 describes the features of the visual programming language Viscuit. Section 4 describes the research methods and the lessons that were conducted. Section 5 presents the results of analyzing the programs created by the children. Section 6 discusses the results, and Section 7 summarizes this report.

2 RELATED WORK

There is a good deal of research on the importance of experiencing computer programming from an early age. Seymour Papert, one of the developers of the educational programming language Logo, said that programming could be a powerful tool for children to express their ideas and help them learn [24]. Nowadays, various tools for teaching programming to young children exist [8, 15, 17, 29, 32, 33], and ways of measuring the effect of learning programming are being explored [13, 20].

A lot of studies show that preschool children are able and motivated to make programs [6, 7, 16, 18, 22, 23]. However, there are few studies about what kind of programs children can make and in which way [28, 36].

Louise et al. researched the programs made by kindergarten children whose ages were 4.4 years to 6.6 years; the mean age was 5.6. They divided the children into cognitive stages [11] and figured out what kind of programs children at each stage could make[10]. In this study, it is confirmed that children at each stage of cognitive development varied significantly in their approaches to programming. When using CHERP [4, 9], which is designed specifically for around 4-6 year old children to control robots built from LOGO robotics construction kits, the pre-operational stage children sometimes showed difficulties in making programs, while the concrete operational stage children could explore operating the robot by themselves.

Marina et al. executed a trial consisting of six lessons [5]. They analyzed the understanding of programming of children through recording their processes and examining the programs made by them using CHERP [4, 9]. The ages of the participants were 4-6 years. In this study, many children showed high understanding over the course of the first half of the curriculum. However, when it came to the difficult part, the number of children who could not understand increased. Specifically, it was the part in which the idea of conditional programs was taught.

Even for elementary school children, the idea of conditional programs seems to be difficult. Cecilia et al. compared preschool and elementary school children when learning computer science concepts through multi-language robot programming [21]. They reported that children aged 5-6 showed difficulties in learning "conditional programs", but they seemed to understand "sequence", "loop", and "parameter". Linda et al. studied programs made from scratch on the Internet [30]. As a result, the "Collide", "Interact", and "Scoring" that are needed to make a conditional programs were used less frequently than "Looks", "Motion", and "Conversate', which could be made without a conditional programs. Therefore, it seems that the concept of conditional programs is difficult for preschool children.

Regarding the programming language adopting the picture rewriting rule, there are few studies about preschool children. We used Viscuit in this research, and Viscuit adopts the picture rewriting rule [1, 3, 25, 31]. We reported that most of the children aged 5 to 6 who took programming lessons using Viscuit at a kindergarten could move the prepared pictures in the proper direction [35, 36].

In almost all studies, developing computational thinking [37, 38] (CT) was put forward as one of the major purposes of teaching programming to preschool children. Abstraction was introduced as the key of CT [38]. As we are going to explain in the following chapter, less abstraction is needed to make programs in Viscuit. However, we consider that children have become familiar with computers through these lessons.

Alexander et al. describe the computational thinking process (CTP) through which people express their ideas using a computer [26]. The CTP consists of 1) problem formulation, 2) solution expression, 3) solution execution and evaluation, and then go back to 1). In addition, they put forward the concept of the computational thinking tool (CTT) that supports and integrates the three stages of the CTP. For CTT, minimizing unnecessary complexity, such as writing long lines of code, is the important thing. We consider that children in this study definitely follow this process as well when they crafted programs without writing code. In process 1) they find the subject matter that they wanted to make and imagine the movement, 2) they make programs, 3) they make sure of the results of their programs and work on other tasks, and then go back to 1).

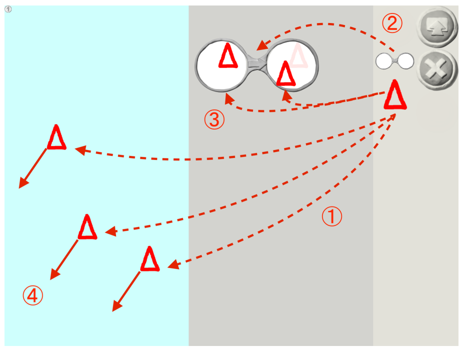

Figure 1: Viscuit interface

We summarize those findings as follows:

- We found plenty of researches focusing on understanding robots but few researches focusing on the picture rewriting rule.
- We realized that conditional programs are difficult for children.
- We consider that children can learn CT without writing complex code through CTP.

We will clarify how children show their understanding and how they adopt the standard concept when using Viscuit through objectively analyzing the programs created by them.

3 VISUAL PROGRAMMING LANGUAGE VISCUIT

Viscuit [12, 33, 34] is a programming language developed by Yasunori Harada, who is the third author. With Viscuit, one can move and change multiple pictures on the screen using picture rewriting rules [1, 3, 25, 31]. Its distinguishing feature is that the program is expressed solely with pictures and their layouts; no letters are used at all. Thus, Viscuit is very suitable for preschool children.

Figure 1 shows how Viscuit programs are created. (1) Pre-supplied or originally drawn pictures are placed on the stage (light blue area), (2) an empty rule (looks like eyeglasses) is placed in the gray middle area, (3) pictures here represented by triangles are dragged inside both circles of the rule, and (4) this instantly makes the pictures move on the stage. In the case of this figure, three triangles move toward the bottom left because the layout of the triangle in each circle shows that the triangle in the right circle has been moved from the position of the triangle in the left circle to the bottom left of the right triangle. The rule made in process (2) represents a change from the properties in the left circle as the before state to the properties in the right circle as the after state.

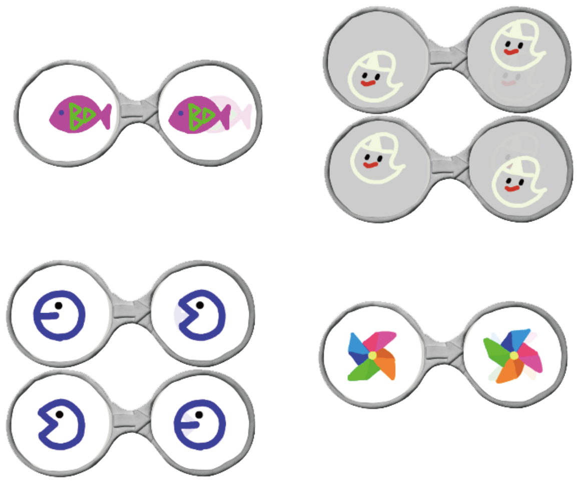

Figure 2: Examples of rules in Viscuit

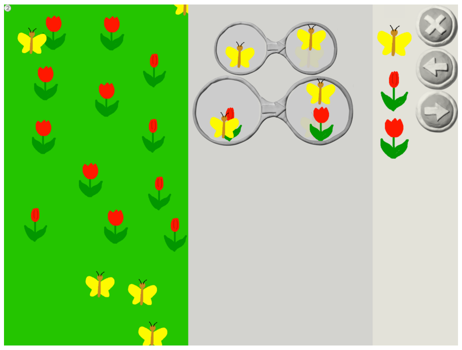

Figure 3: Examples of conditional programs in Viscuit

Figure 2 illustrates how to use the rule in four different ways. Users could make pictures move by putting the same pictures in the left and right circle. The distance between the position of the picture in the left circle and that in the right circle reflects the direction and speed. If the distance is great, the speed increases (Figure 2, upper left). "Random Move" is created by making multiple rules that have the same picture in the left circle (Figure 2, upper right). "Change of Picture" is created when the picture put in each circle is different; the picture in the left circle changes to the picture in the right circle (Figure 2, lower left). "Rotation" is created by rotating one picture (Figure 2, lower right).

Conditional programs is made by putting more than two pictures in the left circle of the rule. The pictures on the stage will move immediately if you put the same picture in the left circle and the right circle. However, if you put multiple pictures in the left circle, it means "if there are pictures positioned in the layout of the left circle, then the pictures of the left circle change to the pictures of the right circle". For example, in Figure 3, in the second rule from top, the left circle of the rule means if a butterfly and a tulip are positioned in the layout of the left circle, the butterfly goes upwards, and the tulip changes to the blooming tulip as the layout of the right circle.

Figure 4: Recital Time Watching Viscuit Land

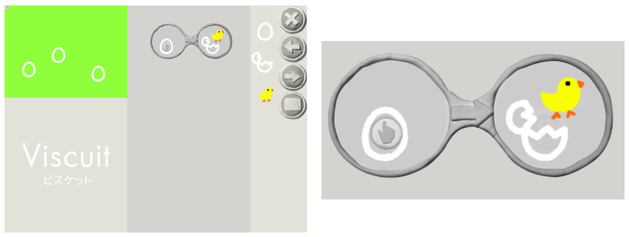

Figure 5: Examples of rules in expert mode

Viscuit has two modes: "novice mode" and "expert mode". In "novice mode", the program will not run unless the number of pictures in the right circle is equal to that in the left circle. The reason for the limitation of the number of pictures is that increasing or decreasing the number of pictures is against the law of conservation of mass. Therefore, this mode is easy for "novices" to understand. In addition, when using "novice mode", many pictures being moved by individual programs can be gathered onto a single large screen called "Viscuit Land" (Figure 4). "We used Viscuit Land a lot in our lessons. In the lessons, children shared the common theme and background provided by the teacher by means of Viscuit Land during the free production time. Each child draws their own pictures, programs the movement of those pictures, and then presses the "save" button to put their pictures onto the shared Viscuit Land.

In "expert mode", a rule can be run even if the number of pictures in each circle is not the same. Therefore, the user can increase or decrease the number of pictures on the stage using the rules. In addition, the user could implement "interaction" by using the finger mark. In Figure 5, the rule means if you touch the egg, it will break, and a chick will appear. In that way, the user could make games by combining various rules (Figure 6). However, we considered that "expert mode" is too difficult for preschool children, so we did not teach this mode to them.

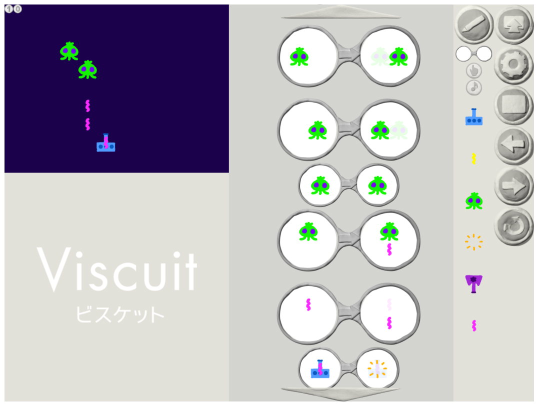

Figure 6: The Shooting game made by Viscuit

4 RESEARCH AND CONDUCTED LESSONS

4.1 Subjects and contents of lessons

The summary of the experiment is as follows.

- Teacher: Kindergarten teacher
- Subjects: 28 5-6 year old children
- Duration of lesson: 40 mins
- Task: to make an introduction for the operetta
- Analysis: programs made by children
- Device: iPad mini
- Year: 2017

The kindergarten obtained consent for the research from all the parents. The backgrounds of the families of the participants were not special; they were ordinary families.

All the lessons lasted 40 minutes, but the composition of the final lesson (L13) was different from that of the previous lessons. L1 to L12 consisted of some practice tasks (Figure 1) in the first half and a free production time based on each practice and review using "Viscuit land" in the second half. L13 contained only free production and the review.

During L13, the children drew pictures with Viscuit with the theme of creating a work introducing the Japanese operetta "The Journey of a Cat Family". It was going to be performed at the kindergarten graduation ceremony for their parents. The story is that a cat family found a vase that indicates the place where a large amount of treasure is hidden, and then the cat family goes on a journey by ship to look for the treasure. However, they encountered a group of Vikings and the Vikings require the vase. Then the son of the family who was holding the vase fell into the sea. The father then jumped into the sea to help the son. Then, all the members of the family jumped into the sea to help the father. After that, the Vikings were impressed by the bonds of the family, so they also jumped into the sea to help the family. Lastly, all of them survived, but they lost the vase. However, they recognized the importance of family.

We created a curriculum to teach programming in Viscuit. Table 1 shows the contents of the lessons. "No." means the number of lessons implemented, and "Date" means the date on which the lessons were implemented in 2017. Through this curriculum, children learned four Viscuit programming techniques, "Straight Move (M)", "Random Move (RM)", "Change of Picture (CP)", and "Rotation (R)" as "Content".

Table 1: Implemented Lessons in School Year 2017

No.	Content	Date
L1-L4	Straight Move (M)	May 11, 25, Jun. 8, 22
L5-L6	Random Move (RM)	Jul. 13, Oct. 26
L7-L9	Change of Picture (CP)	Nov. 9, 30, Dec. 14
L10-L12	Rotation (R)	Jan. 11, 18, 25, 2018
L13	Free Production	Feb. 8

Figure 7: Lesson Examples

As described in Chapter 2, the concept of conditional programs seems to be difficult for kindergarten children to learn [5, 10, 21]. We also felt in the same way in the lessons that we conducted in 2016. In 2016, half of the children, who were first grade elementary school children at that time, could not understand the conditional programs using two pictures (Figure 3). In this series of lessons, we aimed to get almost all of the children to be able to understand programming. Therefore, we omit the conditional programs from our curriculum.

We want them to know at least the following programming features:

(1) When you make programs, then the picture starts to move.
(2) If you make programs in the wrong way, the computer will run in the wrong way.
(3) The rule will be applied to the multiple pictures on the screen.

We consider the first and third features to be abstraction features because they make pictures move indirectly using rules.

In addition, in the previous year, we found that children enjoyed the programming if we did not raise the level of programming every lesson. They enjoyed learning just by changing the appearances of the pictures appearing in each practice. For example, we prepared pictures of animals in the sea to teach Straight Move in the first lesson. In the second lesson, we changed them to animals in the sky. Then, nobody complained about that repetition, and they enjoyed the lesson. Therefore, we decided not to go quickly but keep to a speed such that the slowest children could keep up with us to make all of them understand.

The lesson was conducted by two kindergarten teachers. One of them was in charge of teaching Viscuit, and the other was in charge of the regular class (Figure 7).

Table 2: Information in JSON files

Subject	Focus	Data
File	Size and complexity of programs	No. of pictures placed No. of pictures drawn No. of rules made
Rule	Type of rule	Difference of pictures Rotate or not Existence of Rules Starting same picture
	Speed, direction	Coordinates of pictures placed

4.2 Collecting children's programs

All programs made with the Viscuit system are automatically saved in the JSON [14] format in the Viscuit online server. The programs were recorded when the children pressed the "save" button. After pressing "save", the screen was cleared, the children could start their new work, and the work saved appears on the "Viscuit Land" immediately. The number of works in a lesson varied depending on the child. Every time they saved their work, a JSON file was generated. Every JSON file includes the ID of the tablet from which the server received the Viscuit program. We recorded the tablet ID of each child and used this record to associate each JSON file with the child.

We analyzed programs based on the "Type of rules". In addition, we associated the child with the "Type of rules" he or she made to clarify what kind of programs each child made. In addition to those JSON file data, the authors confirmed what kind of pictures they drew by watching each work. We also associated the pictures and the "Type of rules" to determine whether they could program effectively.

Table 2 shows the information that can be obtained from a JSON file.

5 PROGRAM ANALYSIS

5.1 Programs

The kindergarten children often used multiple rules for a single picture when they made programs. Therefore, we analyzed their works based on each picture. For example, if picture A was being moved using two or more rules, the picture and the rules were counted as one set of rules.

In L13 (the final lesson), the numbers of the various entities are shown below.

- Number of children: 28
- Number of programs (JSON): 128
- Number of rules in each file: 206
- Number of valid rules: 196
- Number of pictures: 233
- Number of sets of rules: 152

5.2 What children expressed in programs

5.2.1 Types of techniques used by children. The number of sets of rules was 152. The 152 sets of rules were categorized according to the five techniques they realize: "Straight Move (M)", "Random Move (RM)", "Change of Picture (CP)", "Rotation (R)", and "Conditional programs (Con)". Those techniques were learned by children throughout the year (Table 1), except for "Conditional programs (Con)", which was not taught, but one child used it. Figure 3 shows the results of the classification. Here, "1 Way" in "Change of Picture" is the rule for a one-way change for changing A to B with no further changes. During the lessons, we taught "Change of Picture" as a means of creating repeating changes (Figure 2). Therefore, we judged a "1 Way" change as a mistake and excluded it from the analysis, although these one-way changes were tabulated. Regarding "Conditional programs", it was surprising that there was a child who applied this technique even though it was not taught.

Table 3: Number of categorized rules

Tech	Number	%	p-value
R	51	33.6%	$p < 0.05$
M	49	32.2%	$p < 0.05$
CP (repeating)	33	21.7%	$p > 0.05$
CP (1 way)	10	6.6%	
RM	7	4.6%	$p < 0.05$
Con	2	1.3%	Not executed

We executed the binomial test on all of the techniques except for "Conditional programs". We assumed that the probability of the technique the children applied was the same as the null hypothesis. Here, we regarded "repeating" and "1 way" as the same technique of "Change of Picture".

We confirmed that "Rotation" and "Straight move" were selected more significantly; however, "Change of Picture" was not applied more significantly. With respect to "Rotation" and "Straight Move", we guessed they were easier to use than "Change of Picture", so the children applied this technique more than "Change of Picture. Regarding "Random Move", we assumed that children did not use "Random Move" enough to express their ideas in L5 and L6 (Table 1). This rule was only taught for two days although other content was taught more than three times. Also, they were on summer vacation between these two days.

Table 4 shows which pictures were used as motifs in each of the 152 sets of rules. Of the pictures adopted as motifs, "Cat" was the most common, followed by "?", "Wave", "Vase", and "Ship". "?" means that the authors could not identify the picture. "Others" contains those pictures that we could identify, but the occurrence was only once or twice. We judged that many of the motifs were selected intentionally because those motifs actually appeared in the operetta. We marked with "*" the name of pictures that appeared in the operetta. On the other hand, there were 32 pictures classified as "?". This represents 21.05% of the 152 sets of rules. In addition to the total number, the number of sets of rules for each picture is shown. The most adopted technique for each set is marked in Table 4.

We found that the number of techniques applied to each picture was different. We presumed that the reason for the differences between those pictures is caused by the properties of each picture. For example, "Wave" would undulate and "Ship" would go straight. We analyzed the details of each set of rules separately for

Table 4: pictures and Techniques

Picture	Total	M	R	CP	R	CON
Cat*	38	12	1	13*	7	0
?	32	10	0	6	12*	0
Wave*	21	4	1	0	16*	0
Vase*	13	3	1	4*	4*	0
Ship*	12	6*	3	1	2	0
Star	4	0	0	0	4*	0
Candy	3	0	0	1	2*	0
Fish*	3	3*	0	1	0	0
Other	26	13*	1	8	4	2

Table 5: Pictures and Directions

Direction	Picture	Total	U	D	L	R
Yes	Ship*	6	0	0	6	0
	Fish*	4	0	0	1	3
	Cat*	3	0	0	1	2
	Vehicle	2	0	0	1	1
	Human	1	0	0	0	1
	Wave*	1	0	0	1	0
None	?	10	1	1	5	3
	Cat*	9	1	0	3	5
	Vase*	3	0	0	0	3
	Part of body	2	0	0	0	2
	Wave*	2	0	0	1	1
	Other	6	4	0	1	1

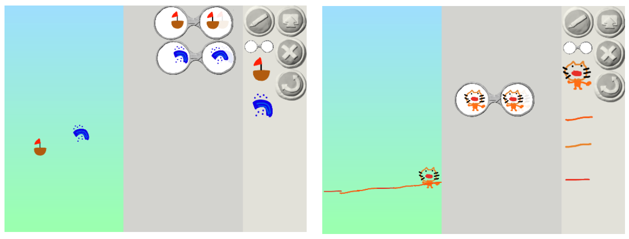
Figure 8: Examples of Linear Move

"Straight Move", "Random Move", "Change of Picture", "Rotation", and "Conditional programs".

5.2.2 *Straight move.*

There were a total of 49 rules for "Straight Move". Table 5 shows the associated pictures and directions of movement for those rules. "U" stands for "Upwards", "D" stands for "Downwards", "L" stands for "Leftwards", and "R" stands for "Rightwards".

Some of the pictures drawn by children had direction. We could recognize whether the particular picture had direction by considering whether 1) the direction of the head and the direction of movement match, 2) we could determine whether the direction was vertical or horizontal, and 3) we could determine the direction from the relationship of the picture with other pictures.

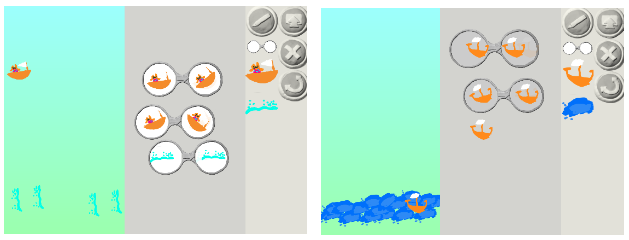
Figure 9: Examples of Random Move

Table 6: Pictures and Changing parts

Picture	Total	Expression	Whole	Color	Other
Cat*	13	10	2	1	0
?	6	0	5	1	0
Vase*	4	0	0	3	1
Face	2	2	0	0	0
Others	8	0	3	5	0

Figure 8 shows the examples. In the rule containing the picture of ship in the left figure, we applied the way of 2) to recognize the direction. The ship must go right or left because it was written from the point of view of the side. In the right figure, we applied the way of 3) to recognize the direction. We could determine the direction of the picture by using the written ground line. We could recognize that it must be horizontal. For 1), the rule like shown at the upper left in Figure 2 was applied.

Of the 49 pictures, 17 indicated the direction of movement, while the remaining 32 pictures did not indicate any direction of movement. In fact, all 17 directional pictures were moved in the proper direction.

5.2.3 *Random move.*

"Random Move" had a small number of adopted sets of rules, seven in total. Looking at each work in detail, three of the seven were movement of ships swaying and rolling, with the waves moving (Figure 9). We suppose that the swaying and rolling ships represent the scene in the operetta, in which the cat family experienced big waves. The other four sets included a picture of the cat, wave, ghost, and vase. The cat and the ghost were made of multiple linear movement rules. The wave and the vase were accompanied by a rotational movement. When we taught "Random Move" during the lessons, we used only "Straight Move". We did not teach programs that combine "Random Move" and "Rotation". In spite of this situation, we found that some children made programs combining techniques they had learned by themselves.

5.2.4 *Change of picture.*

"Change of Picture" had a total of 42 sets of rules. Nine of these were 1-way changes, with picture A changing to picture B. Thirty-three were repetition: changing from A to B and then from B to A.

Table 6 shows the types of pictures to which "Change of Picture" was applied. The motifs used most were "Cat" and "?". When we checked how the "Cat"s were changed, the number of expressions was the most, at ten.

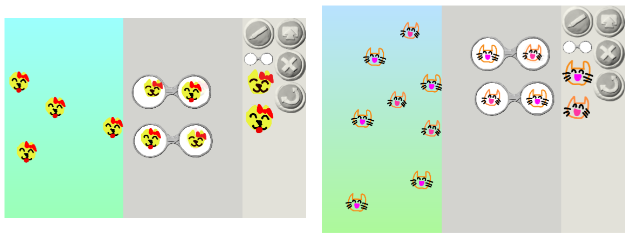
Figure 10: Example of cat changing face

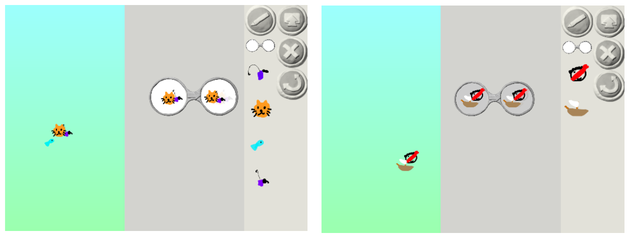
Figure 11: Example of cat conditional programs

Table 7: Pictures and Rotations

Picture	Total	Same Place	Small	Large
Wave*	16	6	4	6
?	12	6	3	3
Cat*	7	4	3	0
Star	4	1	2	1
Vase*	4	4	0	0
Candy	2	1	1	0
Ship*	2	1	0	1
Other	4	3	1	0

Figure 10 shows examples of sets of rules in which the “Cat” changes its facial expression. Two similar pictures were drawn, and the facial expression was changed by changing certain parts of the pictures. We suppose that these “Cat”s were crafted to express the cat family who appear in the operetta.

In regard to “Vase”, the number of “Vase”s changing their color was 3, and 1 “Vase” was changing its shape. In the operetta, there is a scene in which the cat father found a map of treasure on the surface of the vase when he cleaned the dirty vase. We supposed that the “Vase” changing color means that the “Vase” has been cleaned.

5.2.5 Rotation. There were 51 sets of rules for the “Rotation” (Table 7). There were 3 types of rotation: large circle “Rotation”, small circle “Rotation”, and rotation in the same place. “Wave” was adopted for “Rotation” the most, 16 times. The number of large circles rotating was the highest, the same as those rotating in the same place. In addition, other pictures did not have the large circles rotating like “Wave”. We speculated that the rotating “Wave” in the large circle represented the big waves that the cat family encountered in the operetta.

5.2.6 Conditional programs. We found that two works applied “Conditional programs”. One of them showed a cat carrying a fishing rod, and the other showed a Viking riding a ship (Figure 11). These works were composed of two pictures. These two pictures move together but do not move separately. They move only when the two pictures are put as the layout in the left circle of the rule. They move because the position of the pictures in each of the two circles is the same. Both of them were made by the same child.

Table 8: Group details

Num of Tech	Total	Technique	Num
2	16 (57.14%)	M+R	8 (28.57%)
		R+CP	4 (14.29%)
		M+CP	2 (7.14%)
		RM+R	1 (3.57%)
		CP+Con	1 (3.57%)
3	6 (21.43%)	M+CP+R	5 (17.84%)
		M+CP+RM	1 (7.14%)
1	4 (14.29%)	M	1 (3.57%)
		RM	1 (3.57%)
		CP	2 (7.14%)
4	2 (7.14%)	M+CP+RM+R	2 (7.14%)

5.3 Numbers and types of programs made by children

We counted the number of sets of rules created by each child. All children made at least one work. The highest number of rules was 13 sets. The average number of sets of rules made by children was 5.43 sets.

Table 8 shows the number of children showing which techniques they applied. “M” stands for “Straight Move”, “RM” stands for “Random Move”, “CP” stands for “Change of Picture”, “R” stands for “Rotation”, and “Con” stands for “Conditional programs”.

16 children used two techniques, especially “Straight Move” and “Rotation”. Four children used only one technique. Two children used all techniques that were taught. We assume that these results occurred because “Straight Move” and “Rotation” were the easiest techniques.

We confirmed that many works adopted the programs for pictures effectively. However, the number of sets of rules varied greatly among children. Therefore, there was a possibility that a small number of children who were good at crafting programs in Viscuit might have a large effect on the aggregation.

Therefore, we defined the following sets of rules as the sets of rules expressing their ideas effectively.

- Rules that match the direction in “Straight Move”: 17 pics.
- Rules that change the facial expression of “Cats” in “Change of Picture”: 10 pics.
- Rules that change appearance of “Vase” in “Change of Picture”: 3 pics

Table 9: Number of techniques used by each child

Children	No. of techniques	M	RM	CP	R	Con
A	1		1			
B	2	2			1	
C	1	2				
D	1	3				
E	1				3	
F	2	1			1	
G	1	1				
H	1			1		
I	2	1			3	
J	2	1			1	
K	3		1	2	1	
L	1				1	
M	1	1				
N	2			3		2
O	1			1		
P	1			1		
Q	1			2		
R	2	2		1		
S	2			1	1	
T	2		1		1	
U	2	1			1	
V	1				1	
W	1	1				
X	2	1			1	
Total		17	3	12	16	2

- Rules that make the ship sway and roll in "Random Move": 3 pics.
- Rules of the waves in "Rotation": 16 pics.
- Rules that apply "Conditional programs": 2 pics.

Then, we calculated the number of each set. We found that almost all children contributed to the number of these works. Twenty-four of the 28 children (85.71%) were able to use the above expressions (Table 9). Also, 13 children used one type of learned techniques, 10 children used two, and 1 child used three.

When we focused on 4 of the 28 children, who did not contribute the number of the sets of rules expressing their idea effectively(Table 9), we found that they made programs successfully. For example, all of the four children drew something that was related to the operetta. They succeeded in making a rule applying "Change of Picture". Therefore, we could say they could understand programming and the purpose of the lesson.

5.4 From Childcare Diary

The teacher of the kindergarten who executed these lessons kept a childcare diary. In the diary, she wrote the following:

- I think this was the best lesson. Children succeeded in making programs that showed their own character.
- They were pleased that the theme was operetta. Hence, their motivation was increased when they heard that the work would be used to introduce the operetta that will be attended by their parents.
- There were some children who wrote all of the roles. Others wrote only the role that he or she plays. In addition, some children wrote saying "this picture may not appear in the story, but it will appear in the world of the story".

From this diary, we could determine that making programs using Viscuit stimulated the children's imagination. They were very motivated to express their ideas.

6 DISCUSSION

Throughout the year, kindergarten children had lessons using Viscuit programming language. They were taught how to use techniques such as "Straight Move", "Random Move", "Change of Pictures", and "Rotation". We analyzed how children used each technique to express their idea in the final lesson.

The total number of techniques used by all children was calculated. The ratio of "Random Move" was 4.6%, and it was applied significantly less than the other techniques (Figure 3). In addition, "Straight Move" and "Rotation" were applied significantly more. Those results were statistically significant based on the binomial test. On the other hand, "Change of Picture" was not applied significantly more. We do not think "Random Move" is more difficult than the other three techniques, but children might need more time to apply it in a series of lessons. It can be said that programming with "Change of Pictures" is slightly more difficult for them compared to programming with "Straight Move" and "Rotation".

Some of the children made programs using techniques that were not taught. "Random Move" was applied with "Rotation" to express a swaying boat despite the fact that we taught them separately. In addition, one child seemed to find out how to use "Conditional programs". This would imply that children who are on the concrete operational stage could explore unknown techniques by themselves without being taught. This is consistent with Louise's study [10].

We confirmed that all of the children could make programs. This seemed to match the previous results [10, 19]. In addition, they applied their way of programming to express their idea validly. Regarding the direction of movement of the pictures, we confirmed that the children made programs in the proper direction again [36]. For "Random", "Change of Picture", and "Rotation", we confirmed that most of them were applied validly to express the operetta. Children could successfully make programs using the picture rewriting rules.

All children could make more than one set of rules, and the average number was 5.42. 85.7% of them applied more than two techniques to their works (Table 8). In this process, we think children went through CTP [26] because they must have considered which techniques to apply to which pictures.

When we looked at the technique usage in detail, we found that almost all children, 85.7% of them, contributed to the sets of rules expressing their ideas effectively (Table 9). In addition, the rest of them also made works that were related to the operetta. We believe that our curriculum was appropriate for them. All of them most likely understood programming in Viscuit. It means we succeeded in teaching what we want them to learn.

Considering the childcare diary written by the kindergarten teacher, each child seemed to work on the lesson showing one's character, and they were motivated. We confirmed that programming could empower children's learning individually, which is in accordance with what Papert said. He said that programming could be a powerful tool for them to express their ideas [24].

7 CONCLUSION

This study analyzed how kindergarten children craft programs using Viscuit by adopting the picture rewriting rules. In the experiment, children effectively crafted programs by applying various techniques to their own pictures. We conclude that the children could make programs using Viscuit.

In addition, it is very likely that kindergarten children understand the techniques used in their programs. They internalize them, selected the learned techniques that matched the pictures they drew, and expressed their ideas. In general, we can say that programming helps children to express their ideas. This conclusion could apply to all children attending this research.

In this study, the number of participants was small. We should execute the same experiment with more children to confirm the results we discuss in this research.

As future work, we would like to study whether the same results could be achieved if we let children make programs with other themes. In addition, in these lessons, we omit teaching conditional programs; however, we should seek the way to teach conditional programs.

REFERENCES

[1] Michael Anderson and George Furnas. 2010. Relating two image-based diagrammatic reasoning architectures. In *International Conference on Theory and Application of Diagrams*. Springer, 128–143.

[2] Michal Armoni and Judith Gal-Ezer. 2014. Early Computing Education: Why? What? When? Who? *ACM Inroads* 5, 4 (Dec. 2014), 54201359. https://doi.org/10.1145/2684721.2684734

[3] B. Bell and C. Lewis. 1993. ChemTrains: a language for creating behaving pictures. In *Proceedings 1993 IEEE Symposium on Visual Languages*. 188–195.

[4] Marina U Bers. 2010. The TangibleK Robotics program: Applied computational thinking for young children. *Early Childhood Research & Practice* 12, 2 (2010), n2.

[5] Marina Umaschi Bers, Louise Flannery, Elizabeth R. Kazakoff, and Amanda Sullivan. 2014. Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers Education* 72 (2014), 145 – 157. https://doi.org/10.1016/j.compedu.2013.10.020

[6] L. G. Caguana Anzoàtegui, M. I. Alves Rodrigues Pereira, and M. del Carmen Solís Jarrìn. 2017. Cubetto for preschoolers: Computer programming code to code. In *2017 International Symposium on Computers in Education (SIIE)*. 1–5.

[7] D.H. Clements. 1986. Effects of Logo and CAI enviroments on cognition and creativity. *Journal of Educational Psychology* 78, 4 (1986), 309–318. https://doi.org/10.1037/0022-0663.78.4.309

[8] Cubetto 2019. *Cubetto A toy robot teaching kids code & computer programming.* Retrieved Mar 27, 2019 from https://www.primotoys.com/

[9] DevTech 2011. *DevTech Research Group.* Retrieved Mar. 5, 2020 from https://ase.tufts.edu/devtech/index.html

[10] Louise P. Flannery and Marina Umaschi Bers. 2013. Let's Dance the "Robot Hokey-Pokey!"Children's Programming Approaches and Achievement throughout Early Cognitive Development. *Journal of Research on Technology in Education* 46, 1 (2013), 81–101. https://doi.org/10.1080/15391523.2013.10782614 arXiv:https://doi.org/10.1080/15391523.2013.10782614

[11] J. H. Flavell. 1996. Piaget's Legacy. Psychological Science. *Journal of Research on Technology in Education* 7, 4 (1996), 2002013203. https://doi.org/10.1111/j.1467-9280.1996.tb00359.x

[12] Y. Harada and R. Potter. 2003. Fuzzy rewriting: soft program semantics for children. In *IEEE Symposium on Human Centric Computing Languages and Environments, 2003. Proceedings. 2003.* 39–46.

[13] Mia Heikkilä and Linda Mannila. 2018. Debugging in Programming as a Multimodal Practice in Early Childhood Education Settings. *Multimodal Technologies and Interaction* 2, 3 (2018). https://doi.org/10.3390/mti2030042

[14] Json 2018. *JSON - Wikipedia.* Retrieved Apr. 5, 2018 from https://en.wikipedia.org/wiki/JSON

[15] Kalliopi Kanaki and Michail Kalogiannakis. 2018. Introducing fundamental object-oriented programming concepts in preschool education within the context of physical science courses. *Education and Information Technologies* 23 (2018), 2673–2698.

[16] Marina Kazakoff, Elizabeth;Bers. 2012. Programming in a robotics context in the kindergarten classroom: The impact on sequencing skills. *Journal of Educational Multimedia and Hypermedia* 21, 4 (November 2012), 371–391.

[17] Kodu 2020. *KODU GAME LAB COMMUNITY.* Retrieved Mar. 5, 2020 from https://www.kodugamelab.com/

[18] Kaitlyn D Leidl, Marina Umaschi Bers, and Claudia Mihm. 2017. Programming with ScratchJr: a review of the first year of user analytics. In *Conference Proceedings of International Conference on Computational Thinking Education.* 116–121.

[19] Linda Mannila, Valentina Dagiene, Barbara Demo, Natasa Grgurina, Claudio Mirolo, Lennart Rolandsson, and Amber Settle. 2014. Computational Thinking in K-9 Education. In *Proceedings of the Working Group Reports of the 2014 on Innovation Technology in Computer Science Education Conference* (Uppsala, Sweden) *(ITiCSE-WGR '14)*. Association for Computing Machinery, New York, NY, USA, 1201329. https://doi.org/10.1145/2713609.2713610

[20] Eva Marinus, Zoe Powell, Rosalind Thornton, Genevieve McArthur, and Stephen Crain. 2018. Unravelling the Cognition of Coding in 3-to-6-Year Olds: The Development of an Assessment Tool and the Relation between Coding Ability and Cognitive Compiling of Syntax in Natural Language. In *Proceedings of the 2018 ACM Conference on International Computing Education Research* (Espoo, Finland) *(ICER '18)*. Association for Computing Machinery, New York, NY, USA, 1332013141. https://doi.org/10.1145/3230977.3230984

[21] Cecilia Martinez, Marcos J. Gomez, and Luciana Benotti. 2015. A Comparison of Preschool and Elementary School Children Learning Computer Science Concepts through a Multilanguage Robot Programming Platform. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (Vilnius, Lithuania) *(ITiCSE '15)*. Association for Computing Machinery, New York, NY, USA, 1592013164. https://doi.org/10.1145/2729094.2742599

[22] Leonel Morgado, Rosa Cristóvão Morgado, Maria Cruz, and Ken Kahn. 2005. Embedding Computer Activities into the Context of Preschools. (01 2005).

[23] Stamatios Papadakis, Michail Kalogiannakis, and Nicholas Zaranis. 2016. Developing Fundamental Programming Concepts and Computational Thinking with ScratchJr in Preschool Education: A Case Study. *Int. J. Mob. Learn. Organ.* 10, 3 (Jan. 2016), 1872013202. https://doi.org/10.1504/IJMLO.2016.077867

[24] Seymour A. Papert. 1993. *Mindstorms: Children, Computers, And Powerful Ideas.* Basic Books.

[25] Alexander Repenning. 1993. *Agentsheets: A Tool for Building Domain-Oriented Dynamic, Visual Environments.* Technical Report.

[26] Alexander Repenning, Ashok Basawapatna, and Nora Escherle. 2016. Computational thinking tools. In *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 218–222.

[27] Kathryn M. Rich, Carla Strickland, T. Andrew Binkowski, and Diana Franklin. 2019. A K-8 Debugging Learning Trajectory Derived from Research Literature. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 7452013751. https://doi.org/10.1145/3287324.3287396

[28] Kathryn M. Rich, Carla Strickland, T. Andrew Binkowski, Cheryl Moran, and Diana Franklin. 2017. K-8 Learning Trajectories Derived from Research Literature: Sequence, Repetition, Conditionals. In *Proceedings of the 2017 ACM Conference on International Computing Education Research* (Tacoma, Washington, USA) *(ICER '17)*. Association for Computing Machinery, New York, NY, USA, 1822013190. https://doi.org/10.1145/3105726.3106166

[29] ScratchJr 2018. *ScratchJr.org: Coding for Young Children.* Retrieved Oct 1, 2018 from https://www.scratchjr.org/

[30] Linda Seiter and Brendan Foreman. 2013. Modeling the Learning Progressions of Computational Thinking of Primary Grade Students. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (San Diego, San California, USA) *(ICER '13)*. Association for Computing Machinery, New York, NY, USA, 59201366. https://doi.org/10.1145/2493394.2493403

[31] David Canfield Smith, Allen Cypher, and Jim Spohrer. 1994. KidSim: Programming Agents without a Programming Language. *Commun. ACM* 37, 7 (July 1994), 54201367. https://doi.org/10.1145/176789.176795

[32] Amanda A Sullivan, Marina Umaschi Bers, and Claudia Mihm. 2017. Imagining, playing, and coding with KIBO: using robotics to foster computational thinking in young children. *Siu-cheung KONG The Education University of Hong Kong, Hong Kong* 110 (2017).

[33] Viscuit 2020. *Viscuit.* Retrieved Mar. 5, 2020 from https://www.viscuit.com

[34] Takeshi Watanabe, Yuriko Nakayama, Yasunori Harada, and Yasushi Kuno. 2018. Programming Lessons for Kindergarten Children in Japan. In *Constructionism2018* (Vilnius, Lithuania). 741–744. http://www.constructionism2018.fsf.vu.lt/file/repository/Proceeding_2018_Constructionism.pdf

[35] Takeshi Watanabe, Yuriko Nakayama, Yasunori Harada, and Yasushi Kuno. 2019. How Can Children Express Their Intentions Through Coding? Analysis of Viscuit Programs in Kindergarten. In *Proceedings of the 2019 ACM Conference on*

Innovation and Technology in Computer Science Education. 326–326.

[36] Takeshi Watanabe, Yuriko Nakayama, Yasunori Harada, and Yasushi Kuno. 2020. Analyzing Understanding of the Direction in Viscuit Programs Crafted by Kindergarten Children. *IPSJ Transactions on Computers and Education* 6, 1 (Feb 2020), 28–39. https://ci.nii.ac.jp/naid/170000181714/"

[37] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.

[38] Jeannette M Wing. 2014. Computational thinking benefits society. *40th Anniversary Blog of Social Issues in Computing* 2014 (2014).