Quantitative Learning Effect Evaluation of Programming Learning Tools

Abstract— **Children can learn programming using different tools. Understanding how the characteristics and features of each tool impact the learning effect will enhance learning. However, the impact of specific tools on the learning effect is unclear. In this study, we conducted a workshop to evaluate the characteristics and features of six tools on the learning effect. Our study reveals that the learning effect clearly differs between the six tools.**

Keywords—Programming Education, Programming Learning, Programming Learning Tools

I. INTRODUCTION

Various learning tools exist for first-time learners of programming [1] [2]. Each tool has unique characteristics and features. To enhance the learning effort, a tool must be appropriately selected based on the learning purpose. Several studies have evaluated various tools, but the learning effects due to the characteristics and features of a given tool have yet to be sufficiently examined.

To solve the aforementioned issue, this research investigates the following Research Questions (RQs):

- RQ1: Is there a difference in characteristics and features between programming tools?
- RQ2: Does the programming tool influence the learning effect?
- RQ3: Is there a relation between the characteristics and features of a tool and the learning effect?

RQ1 determines whether each tool has unique characteristics and features. The most appropriate tool for the intended purpose can be selected based on the desired characteristics and features. Therefore, RQ1 should enhance the effectiveness of applying tools.

Because the tool should impact the learning effect, RQ2 evaluates the influence of each tool on the learning effect. RQ3 will elucidate the learning effects based on the characteristics and features of each tool. Understanding the learning effect from these perspectives will help select the appropriate tool based on learning objectives and goals.

The rest of this paper is organized as follows. Section 2 describes the background. Section 3 describes the survey tools, while section 4 overviews the experiments. Section 5 shows the evaluation results of each tool. Section 6 addresses the RQs and threats against validity. Section 7 introduces a relevant study, while section 8 provides the conclusion and future work.

II. BACKGROUND

Programming learning for beginners has been conducted using various learning tools. As examples, Scratch [3][4] is used in a visual programming language, while CodeCombat [5] and Minecraft Education Edition [6] exist in game software. These tools have different characteristics, including program expression and programming method. For example, a program expression can be text, visual, etc. A previous study on multimedia learning revealed that learner recognition and learning effects differ from the viewpoint of text expression and image expression [7].

In addition, these tools differ widely from the developers' thoughts, purposes of use, and learning objectives. Although many researchers have investigated programming learning tools (e.g., evaluation with a single tool [8] and comparisons between text and visual languages [9]), few studies have compared programming learning tools in multiple fields. Therefore, the kinds of learning effects due to the characteristics and features of the programming learning environment are unknown.

In this research, we evaluate tools with three different programming methods [visual programming languages, game software, and physical tools (tangible and unplugged)] in the same framework using a workshop.

III. PROGRAMMING LEARNING TOOLS

We selected six tools that are commonly available in Japan.

A. Scratch (Sc)

Scratch (Fig. 1) is a visual language used to create stories, games, and animations. This globally popular tool was developed by MIT Media Laboratory. Some studies [8][10] have used this tool.

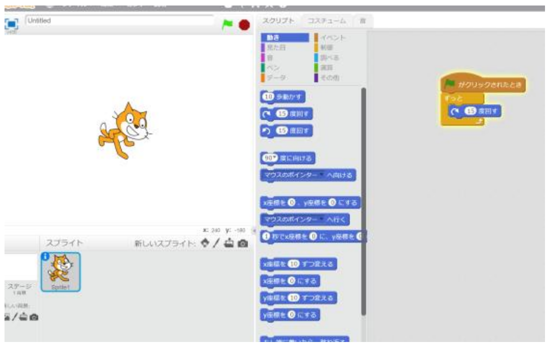


Fig. 1. Scratch

B. *Viscuit (Vi)*

Viscuit (Fig, 2) is a Visual Programming Language and Environment. This tool was developed by Digital Pocket in Japan. It can control the written Illustration using special programming called "glasses".

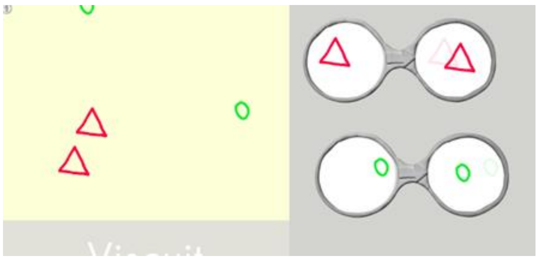
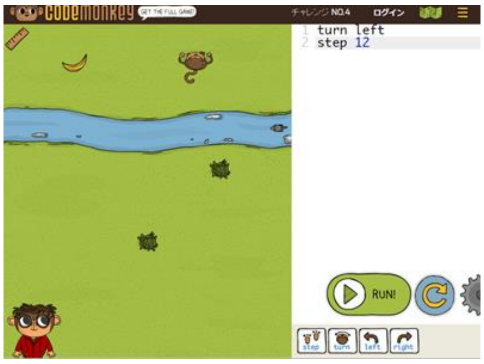


Fig. 2. Viscuit

C. *CodeMonkey (CM)*

CodeMonkey (Fig. 3) is game software used to program the behavior of a monkey collecting bananas. This game uses a programming language called coffee script.



```
turn left
step 12
```

Fig. 3. CodeMonkey

D. *Lightbot (Li)*

Lightbot (Fig. 4) is game software used to program the behavior of a robot to achieve a goal. It teaches the concept of recursion as a "Loop".

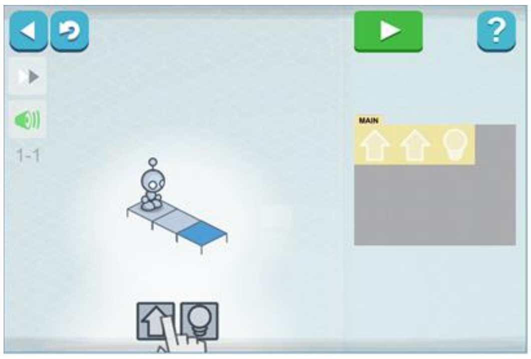


Fig. 4. Lightbot

E. *OSMO Coding (OC)*

Osmo Coding (Fig. 5) is a tangible device. It uses physical blocks for programming to control characters via an iPad application.

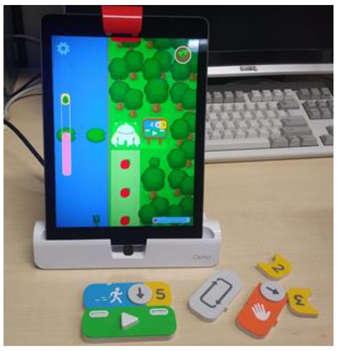


Fig. 5. Osmo Coding

F. *Robot Turtles (RT)*

Robot Turtles (Fig. 6) is a board game in an unplugged tool. The purpose is to create a program to manipulate the turtle and collect jewels.

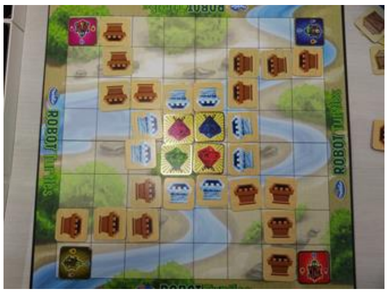


Fig. 6. Robot Turtles

G. *Classification*

These six tools can be divided by characteristic into three fields: visual programming environment (VP), game software (GM), and physical tool (PT). In addition, we qualitatively evaluate the tools based on the taxonomy of Kelleher et al [1]. The results are shown in Table 1.

The visual programming environment uses a visual programming language in a programming method with a drag and drop feature. This feature allows content to be freely created. Viscuit and Scratch are visual programming environments. Their main difference is the expression of code. Scratch is expressed in text, whereas Viscuit is expressed in images.

Game software is software with game elements of Rules/Restrictions, Goals, and Rewards [11][12][13]. Lightbot and CodeMonkey are game software. These tools differ in the expression of code and programming method. Lightbot expresses code in images and programming is by drag and drop. In contrast, CodeMonkey uses text to express code and the programming method is typing the code.

A physical tool refers to a tool that allows programming using physical cards or blocks. OSMO Coding and Robot Turtles are examples. These tools differ in the location of the program execution results. In OSMO Coding, the result of programming is reflected in the software. Therefore, the program works in a virtual space. On the other hand, the execution result of the Robot Turtle is reflected in a piece on a board game. In other words, the program works in real space.

TABLE I. RESULTS OF THE QUALITATIVE EVALUATION

		VP		GM		PT	
		Sc	*Vi*	*CM*	*Li*	*OC*	*RT*
Style of programming	***Procedural***			x	x	x	x
	Object-based	x					
	Object-oriented						
	Event-based	x	x				
Programming constructs	***Conditional***	x		x	x		
	Loop	x		x		x	
	Variables	x		x			
	Parameters	x		x		x	
	Procedures/Methods	x		x	x		x
	Pre and post conditions	x					
	Recursion				x		
Representation of code	***Text***	x		x			
	Pictures		x		x		
	Physical objects					x	x
Construction of programs	***Typing code***			x			
	Assembling graphical objects	x	x		x		
	Selecting/form filling			x			
	Physical objects					x	×
Support to understand programs	***Back stories***			x		x	x
	Debugging						
	Physical interpretation	x		x	x	x	x
	Liveness		x			x	
	Generated examples						
Designing accessible languages	***Limit the domain***	x	x	x	x	x	x
	Select user-centered keywords						
	Remove unnecessary punctuation						
	Use natural language						
	Remove redundancy						
Game elements	***Rule/Restriction***			x	x	x	x
	Goal			x	x	x	x
	Rewards			x		x	

IV. EXPERIMENTS

A. *Experiments*

We focused on the understanding of basic programming concepts (sequential execution, repetition, conditional) and the influence of applied skill (especially, abstraction and problem solving) in a workshop to evaluate the six tools. In addition, we researched the attitudes toward programming via a questionnaire on the attitude towards programming and an eight-point learning comprehension test (programming basics and programming applied test).

B. *Questionnaire and test*

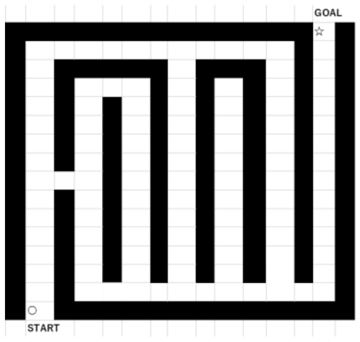
We conducted a questionnaire and a test to analyze the learning effect.

1) Learning comprehension test

The test to investigate the influence of the tool on the understanding of programming consisted of seven questions:

- Sequential: one question
- Repetition: three questions
- Conditional: two questions
- Free description problem: two questions

Figure 7 and 8 show the types of questions asked.



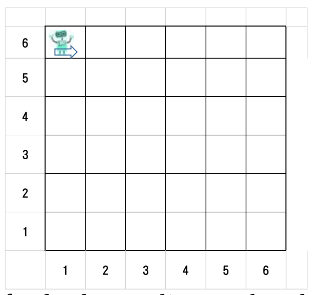
I want to go from the start (○) to the goal (☆).

If you have the following rules, what kind of route do you follow?

Please draw a line in the maze. (Hint: Let's unravel while rotating the paper)

1. If there is a wall on the right hand and there is no wall in front, proceed
2. If there is no wall on the right hand, rotate to the right
3. If there is a wall in front and the right hand, rotate to the left

Fig. 7. Question example



Q1 Please freely draw a line so that the robot passes through all the squares. At first it is facing right.

Q2 Please explain with a simple program why you drew such a line.

Fig. 8. Free description problem

2) Questionnaire about the attitude toward programming

This questionnaire was conducted before and after the workshop to investigate the change in attitude toward programming [fun (Q1A, Q1B), difficulty (Q2A, Q2B), usefulness (Q3A, Q3B), willingness (Q4A, Q4B) and interest (Q5A, Q5B)]. Responses were on a six-stage Likert Scale (1: Strongly disagree, 2: Disagree, 3: Somewhat disagree, 4: Somewhat agree, 5: Agree and 6: Strongly agree). Based on [14], the questionnaire consisted of the following five questions:

- Q1: Do you think programming is fun?
- Q2: Do you think programming is difficult?
- Q3: Do you think programming is usefulness?
- Q4: Do you want to learn programming?
- Q5: Are you interested in programming?

C. *Workshop*

1) Workshop

The workshop system was organized by two to four persons, including lecturers and assistants. The learners were elementary students in grades 3 to 6, except for learners using Robot Turtles. These learners were in grades 1 to 3 in an elementary school where the tool officially was announced as a subject. The teaching materials included online tools, handouts, etc.

2) Schedule of the workshop

The workshop lasted 90 minutes with the following format:

1. Pre-Questionnaire: 2 min;
2. Pre-Test: 5 min;
3. Workshop Time: 75 min;
4. Post-Test: 5 min;
5. Post-Questionnaire: 3 min (+5 additional minutes allowed)

2) Number of students and effective questionnaire responses

Fifty-nine students participated in the workshop [Scratch (10 people), Viscuit (9), CodeMonkey (9), Lightbot (7), OSOMO Coding (16), and Robot Turtles (8)]. The number of valid responses of the test and the questionnaire was as follows:

- Learning comprehension test: 45 people
- Questionnaire of attitude toward programming: 49 people
- Questionnaire on impressions: 49 people

V. Results And Analysis

A. *Learning comprehension test*

1) Overall test results

First, we analyzed the three groups: visual programming environment, game software, and physical tools. Figures 9 – 11 show the learning comprehension test results by group. Each group shows an improved learning comprehension after the workshop. The prior score of each result and the posterior score were tested using the Wilcoxon signed-rank test (confidence interval 95%; p <0.05 indicates a significant difference). Table 2 summarizes the results.

The visual group improves as a whole. The Wilcoxon signed-rank test has a p-value of about 0.08. Although the difference is not significant, the trend indicates that the workshop is effective. However, a few learners have reduced scores after the workshop. One reason for a lower score may be that learners became tired of learning in the visual programming language and stopped taking the test.

The game software group greatly improves in learning. The Wilcoxon signed-rank test has a p-value of about 0.006, which is statistically significant. Game elements are one reason for the significant difference. Because the goal in a game is clear, the students are engaged until the test was complete. However, it is possible that the improved scores are because the problems asked in the test were similar to the game software.

Similarly, the learning effect of the physical tool group improves after the workshop. The Wilcoxon signed-rank test result is not significant with a p-value of about 0.28. The scores of some learners decline after the workshop. The reason is attributed to the difference in the work volume due to physical intervention.

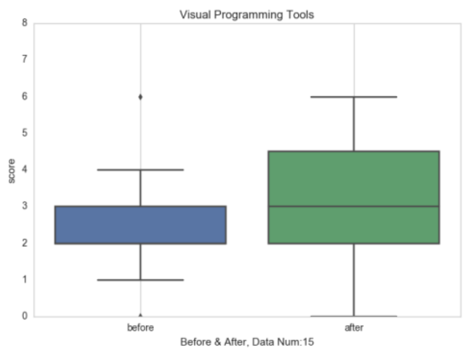


Fig. 9. Results of Visual Programming

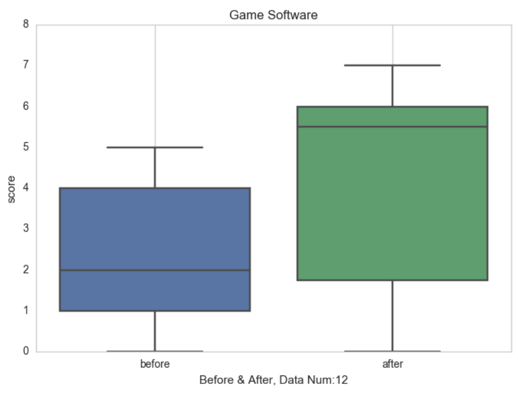


Fig. 10. Results of Game Software

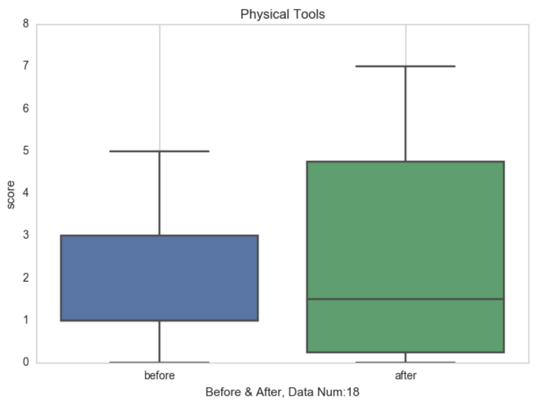


Fig. 11. Results of Physical Software

TABLE II. RESULTS OF THE SIGNIFICANT DIFFERENCE TEST (LEARNING COMPREHENSION TEST)

Category	Statistics	p-value
Visual Programming Tools	17.5	0.0834 *
Game Software	0	0.0059 **
Physical Tools	30	0.2752

[a] ** Significant difference, * Significant trend

2) *Programming applied test*

Two patterns emerge in the responses to the free description questions. The descriptive patterns are either U-shaped (Fig. 12, left) or spiral (Fig. 12, right). Because both are correct due to problem solving, it is possible that learners improve their problem-solving abilities and explanatory skills. The spiral-type can be simply described with a few procedures and components. Therefore, the improvement may be due to an enhanced abstraction ability. It is interesting that only the Viscuit participants responded using a spiral. This suggests that Viscuit may have features not found in the other tools.

None of the learners tackled the explanation of the program prior to the workshop, and only a small number did after workshop. Furthermore, the differences between each tool are not significant. For example, learners felt that they "wanted to proceed until hitting the wall".

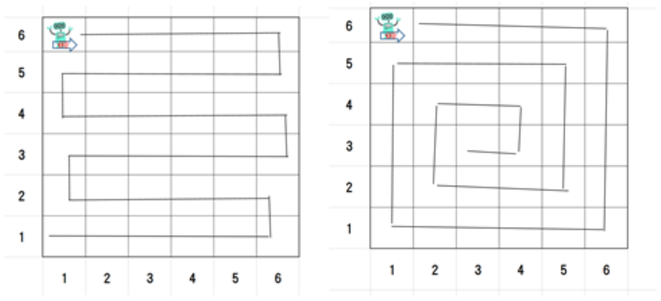


Fig. 12. Results of the Free Description Problem

B. *Attitude toward programming*

Figures 13 – 15 show the results of the questionnaire results on attitude by programming by group. Table 3 shows the Wilcoxon signed-rank test.

If the tool includes game elements, interest in programming improves in the after workshop. Game software is more fun that physical tools with game elements. We also performed a significant difference test using a Wilcoxon signed-rank test. The p-value for interest in the game software group is about 0.06, indicating a significant trend. From a comprehensive viewpoint, game elements make programming feel interesting.

Visual programming languages tend to reduce the difficulty of programming. The degree of difficulty decreases because learners can easily create software by visual programming as it is consistent with the general image of programming. The Wilcoxon signed-rank test shows that the visual programming group has a p-value of approximately 0.09. Therefore, it is a slightly significant trend. Both the game software and physical tools groups felt that programming was more difficult after the workshop. For the game software group, the p-value of the significance test result is about 0.07.

The visual programming language group and the physical tools group indicated that the workshop did not increase the usefulness of programming. However, the game software group felt the usefulness improved after the workshop. This difference may be because game software is instantaneously executed and a concrete result is given. However, the Wilcoxon signed-rank test indicates an insignificant difference by group.

Each group displayed a similar willingness to learn. The Wilcoxon signed-rank test indicates no significant differences. This workshop was adapted with a short introduction, which has a negligible effect on learning willingness. Depending on the tool, some learners reported an impaired learning willingness after the workshop. The reasons need to be further considered.

Each group showed a slight improvement in interest in programming. Although only studying programming for a short time, it seems that the interest improves. However, the Wilcoxon signed-rank test does not confirm a significant difference.

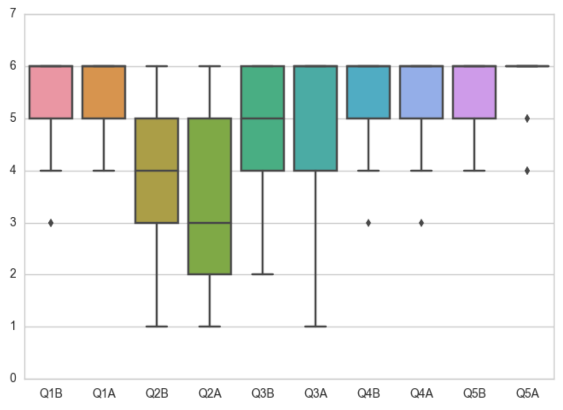


Fig. 13. Results of visual programming language

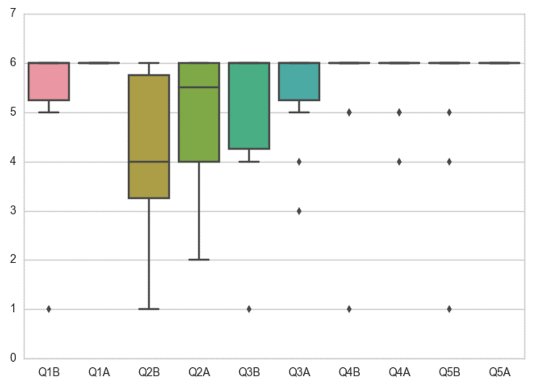


Fig. 14. Results of game software

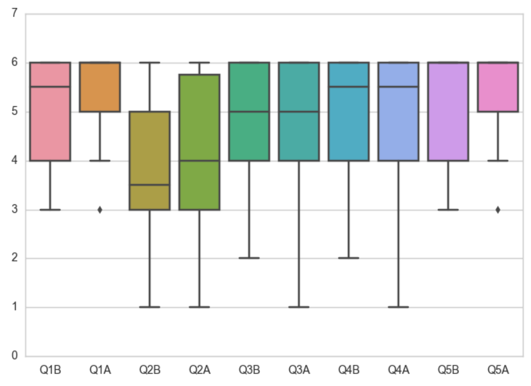


Fig. 15. Results of the physical tool

TABLE III. RESULTS OF THE SIGNIFICANT DIFFERENCE TEST (ATTITUDE TOWARD PROGRAMMING)

	Visual language		**Game Software**		**Physical tool**	
	Statistics	***p-value***	***Statistics***	***p-value***	***Statistics***	***p-value***
Q1	6.000	0.160	0.000	0.059 *	2.000	0.131
Q2	17.500	0.087*	0.000	0.066	26.000	0.522
Q3	11.000	0.608	1.000	0.285	30.500	0.813
Q4	5.500	0.279	7.500	1.000	4.500	0.854
Q5	6.000	0.317	0.000	0.109	2.500	0.157

b. * Significant trend

C. Comparison of the characteristics and features in individual tools

Table 4 overviews the characteristics and features of each tool. In addition, the results of the questionnaire on the impressions about each tool are considered.

1) Scratch

Scratch tends to improve the correct answer rate in the learning comprehension test. Many answers for the free description of learners are U-shaped in the descriptive patterns. Additionally, after the workshop, the "difficulty" for programming is remarkably reduced.

This program method in this tool is to drag and drop a block. Hence, an action is validated immediately after execution. This method is considered to be largely related to the reduction of "difficulty" as assembling graphical objects is a big factor as an element. Furthermore, impressions of "making things" and "making apps" are observed. Thus, learners can quickly visualize movement using illustrations. Furthermore, the high freedom of programming seems to contribute to such impressions.

2) Viscuit

This tool tends to improve the correct answer rate of the learning comprehension test. Both U-shaped and spiral responses are provided in the test of the free description. It is possible that the tool stimulates creativity. The spiral type can be described simply with few procedures and components. Hence, the ability to abstract problems improves after the workshop.

"Moving a picture" and "glasses" are common learner's impressions, which may be because movements with "eyeglasses" are intuitive.

3) CodeMonkey

This tool tends to improve the correct answer rate of the learning comprehension test. Many answers in the free description test are U-shaped. Many learners tried to explain programs in the free description, indicating that the learner thought about and then solved the problem independently. This tool improves explanation skills.

One learner commented, "There were various programs and I learned something very interesting". This tool contains many problems as a collection of problems. The learner adopts a mechanism to progress continuously without a large gap in the difficulty level. This tool seems to lead to continuous enthusiasm and fun. Furthermore, it is easy to express goals and rules of the game elements.

4) Lightbot

This tool tends to improve the correct answer rate of the learning comprehension test because it helps comprehend different programming concepts. Many answers in the free description test of learners are U-shaped descriptive patterns. This tool is a simple puzzle game, which can be operated intuitively using a tablet (including smart phone). The learner sees the program that he or she creates as movements of a robot. Hence, it promotes the understanding of programming concepts.

One learner commented that it is “easier to learn with the feeling of a game”. It is thought that this "game sensation" improves the motivation of learners and promotes the understanding of programming.

5) OSMO Coding

This tool tends to improve the correct answer rate of the learning comprehension test. Many answers in the free description of learners are U-shaped. Although major features are not found for specific matters, each subject is approached in a balanced manner. Because this tool is a tangible device, it is considered effective for learning continuously without decreasing motivation. However, due to the relationship between the physical block and the software element, the workload may increase, causing learners to quit.

In addition, learners noted many impressions of the word "move" such as "move the computer" or "move it as instructed", which are attributed to assembling and programming the blocks.

6) Robot Turtles

The tool tends to improve the correct answer rate of the learning comprehension test. Many answers in the test of the free description of learners are U-shaped. The tool is an unplugged tool, and learners can work in groups. Learning in a group can increase the diversity of knowledge and promote comprehension from the viewpoint of sharing programs created by the students. Cooperation with others also invokes a game element. Impressions suggest that learners think a programming can be optimized, as noted in responses such as a "faster way to go forward”.

TABLE IV. FEATURE TABLE OF THE TOOLS

	Programming constructs						**Attitude toward programming**			
	Sequential	***Loop***	***Conditional 1***	***Conditional 2***	***Free Description (line)***	***Free Description (Description)***	***Fun***	***Difficulty***	***Usefulness***	***Willingness***
Sc			x	x	x			xx		
Vi					x	xx		x		
CM				x	x	x				
Li		x	x	xx	xx					
OC										
RT		x	x							

[c] x = Characteristic and feature, xx = Strong Characteristic and feature

VI. DISCUSSION

A. Answer to the RQs

1) RQ1: Is there a difference in characteristics and features between programming tools?

Each tool exhibits different characteristics and features (e.g., programming method and expression of programming language), confirming RQ1. Table 1 shows the qualitative characteristics and features of the programming tool. Furthermore, as noted by Kelleher et al. [1], some tools have common characteristics and features. For example, visual programming tools employ a programming method using drag and drop. Furthermore, a physical object can be touched by hand with the programming language. In addition, this study used game elements as new elements, and game software has some common elements.

2) RQ2: Does the programming tool influence the learning effect?

Each tool displays its own learning effect. Due to the small sample size, RQ2 should be further investigated. In particular, a difference in the learning effect is observed in the free description problem. However, the influence of each tool on the answer to the free description problem must be further evaluated. This is obvious from the fact that there are two answers (Fig. 12). Furthermore, the questionnaire revealed a difference in "difficulty" for the programming attitude. This is also evident from the results in Fig. 13 - 15. Other attitudes show improvement trends. It has been acknowledged that visual programming tools improve attitudes toward programming [8].

3) RQ3: Is there a relation between the characteristics and features of a tool and the learning effect?

Each tool has learning effects based on its unique characteristics and features, confirming RQ3. The qualitative characteristics and features of the programming tool are listed in Table 1. RQ2 revealed that the learning effect of each tool is different. In particular, factors that influence the learning effect include representation of code and construction of programs. Representation of images and texts affect the recognition in multimedia research [7]. Additionally, the amount of work (e.g., typing the code) in the programming learning environment is affected. It is possible that the difference in this work may influence the learner’s attitude. Furthermore, game elements also influence attitudes toward programming based on the research of Juho Hamari and Veikko Eranti [12]. In addition, Each tool also has characteristics and features by learning effect (Table 4). For example, with Viscuit, spiral type answers are found in the free description problem, suggesting that Viscuit helps cultivate abstraction skills. Therefore, the characteristics and features of each tool may be related to the learning effect. By taking advantage of the unique features, the learning effect may be enhanced according to the intended purpose.

B. Threats to validity

We noted the following threats to validity:

- The population size is small and the number of participants in each tool is biased.
- Some of the test problems are similar to those of the tools.

To address these issues, we will improve the test problems, increase the population size, and enhance the statistical reliability of the data.

VII. Related Works

Kelleher et al. [1] qualitatively investigated and categorized multiple programming environments. However, a quantitative investigation is necessary to assess the characteristics and learning effects. Our research focuses on a quantitative evaluation to clarify the learning effect from characteristics and features.

Paul Gross and Kris Powers [2] summarized evaluations of the programming environment for beginners. Furthermore, they created a rubric to ascertain the quality of their evaluation. They assessed courses of several different environments. In contrast, our research analyzed the tools themselves and investigated the learning effect based on the characteristics and features of the tool. Using both their and our contributions, a more system evaluation may be realized.

VIII. Conclusion

We conducted a quantitative evaluation by a workshop on six programming learning tools. The elements of the classification influence the learning effect. All tools improve the learning comprehension test. However, if the software involves physical elements, the learner may become bored as the workload increases. The three groups (visual programming language, game software and physical tool) show a difference in attitude toward programming. The visual programming language tends to soften programming difficulty. Although tools with game elements tend to make programming more fun, they also increase perceived difficulty of programming.

In the future, we plan to increase the number of tools and the number of learners. We also plan to design a workshop that is independent of the learning tools and lecturers.

References

[1] Caitlin Kelleher and Randy Pausch, “Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers,” ACM Computing Surveys, Vol. 37, No. 2, June 2005, 83–137, DOI: https://doi.org/10.1145/1089733.1089734

[2] Paul Gross and Kris Powers, “Evaluating assessments of novice programming environments” Proceedings of the first international workshop on Computing education research, ACM, 2005, 99-110, Seattle, WA, USA, DOI: https://doi.org/10.1145/1089786.1089796

[3] MIT Media Lab, Scratch - Imagine, Program, Share, at https://scratch.mit.edu/, accessed on 4/13, 2017.

[4] Maloney, John, et al. "The scratch programming language and environment." ACM Transactions on Computing Education (TOCE) 10.4 (2010): 16, DOI: https://doi.org/10.1145/1868358.1868363

[5] CodeCombat Inc., CodeCombat - Learn how to code by playing a game, at https://codecombat.com/, accessed on 4/13, 2017.

[6] Microsoft, Minecraft Education Edition, at http://education.minecraft.net/minecraftedu/, accessed on 6/20, 2016.

[7] Mayer, Richard E. "Multimedia learning." Psychology of learning and motivation 41 (2002): 85-139.

[8] Sáez-López, José-Manuel, Marcos Román-González, and Esteban Vázquez-Cano. "Visual programming languages integrated across the curriculum in elementary school: A two year case study using “Scratch” in five schools." Computers & Education 97 (2016): 129-141.

[9] Daisuke Saito, Hironori Washizaki and Yoshiaki Fukazawa. “Comparison of Text-Based and Visual-Based Programming Input Methods for First-Time Learners”. JITE:Research , Volume 16 , 2017. pp 209 – 226

[10] Resnick, Mitchel, et al. "Scratch: programming for all." Communications of the ACM 52.11 (2009): 60-67.

[11] Katie Seaborn and Deborah I. Fels, “Gamification in theory and action: A survey,” Int. J. Human-Computer Studies 74, 2015, pp. 14-31.

[12] Juho Hamari and Veikko Eranti, “Framework for Designing and Evaluating game Achievements,” Authors & Digital Games Research Association DiGRA, 2011.

[13] Juul, Jesper. Half-real: Video games between real rules and fictional worlds. MIT press, 2011.

[14] Saito, Daisuke, Hironori Washizaki, and Yoshiaki Fukazawa. "Analysis of the learning effects between text-based and visual-based beginner programming environments." Engineering Education (ICEED), 2016 IEEE 8th International Conference on. IEEE, 2016.