International Journal of Learning Technologies and Learning Environments
International Institute of Applied Informatics
2023, Vol. 6, No. 1, IJLTLE730

Development of a Programming Learning Support System with Functions for Switching Display Language and for Showing Students' Learning Status to Teachers

Ryo Tokimatsu [*], Naoya Kamita [*],
Tetsuo Tanaka [*], Kazunori Matsumoto [*]

Abstract

For the education of beginner programmers, the visual programming environment Scratch and the Japanese programming language Nadeshiko are attracting a lot of interest. These are excellent tools for learning programming. However, in order to program in a new language as it is used in practice, it is necessary to learn new language-specific syntax, built-in functions, coding styles, and so on. It can be difficult to apply the knowledge learned in a visual programming environment to real-world programming situations. We propose a programming learning system that has a language switching function that switches the display of blocks of code to another programming language, and a function that executes the program in a visual form. The system enables learners to study efficiently while comparing a language they know with one that they do not, and to intuitively grasp the behavior of a particular program.

Keywords: Visual Programming, Block based, Switching Display Language, Learning Support System, Operation Logs, Blockly

1 Introduction

With the progress of ICT, the demand for software development is increasing. Along with this, the demand for programming education is also increasing, and education related to programming is being strengthened in elementary schools, junior high schools, and high schools. In addition, students' motivation to learn programming is on the rise. According to a survey of elementary and junior high school students regarding programming education conducted by the MMD Institute, 79.6% of elementary and junior high school students want to learn programming [1].

As a programming learning environment for beginners, Scratch [2] [3], Swift Playgrounds [4] and Nadeshiko [5] [6] are attracting a lot of interest. Scratch is a visual programming environment

[*] Kanagawa Institute of Technology, Kanagawa, Japan

that runs on the Web. Swift Playgrounds is an application for iPad and Mac that makes learning Swift interactive and fun. In a visual programming environment such as Scratch or Swift Playground, programming is facilitated by interacting with parts of a program called blocks using the mouse, and it is possible to acquire programming thought processes without accurately memorizing the syntax of any particular programming language [7]. [8] [9] [10]. Nadeshiko is an application that runs on Windows and the Web. You can program in Japanese, and you can learn programs while internalizing the meaning of the source code in Japanese.

These tools are good for learning programming thought processes, but in actual application development, neither visual programming environment nor Japanese programming language are rarely used, and text-based programming languages such as C, Python, and JavaScript are used instead[11][12]. Therefore, it is difficult to utilize the knowledge learned in a visual programming environment such as Scratch as it is actually applied in application development[13][14][15]. It is difficult for students to apply what they have learned in Japanese programming language Nadeshiko to programming in English based programming languages because it is necessary to learn new features such as language-specific syntax, built-in functions, and coding style.

In consideration of this background context, in this research, the authors have developed a visual programming environment for the purposes of supporting programming learning in a text-based language. This system has a language switching function that switches the display of blocks of code to another programming language. This allows learners to learn language-specific syntax while training programming thought processes.

This paper is the extended version of Reference[16]. The authors extended the functions of the system reported in Reference [16] and improved its usability. In addition, we conducted additional experiments on the learning effect and investigated the relationship between the number of language switching operations and the learning effect.

Chapter 2 describes current issues and the approach taken in this research. The function of the learning support system is described in Chapter 3, and its implementation is described in Chapter 4. An evaluation based on a trial experiment is described in Chapter 5.

2 Current Issues and Approach Taken in This Research

2.1 Support for language understanding by switching the display language

Learning in a visual programming environment such as Scratch presents a problem in that it is difficult to use such a system to acquire knowledge of text-based languages like Java and Python. This research solves this problem by means of a function for switching between display languages, as shown in Figure 1.

With this function, users can switch the language displayed in a block of code at any time during programming. This allows them to learn language-specific syntax while training programming thought processes.

As regards the switching of blocks of code, in addition to "one-to-one switching" that swaps one instruction block with one other instruction block, "one-to-many switching" that swaps one instruction block with multiple instruction blocks is also possible. Thus, the system makes it possible to learn while switching between languages with different abstractions.

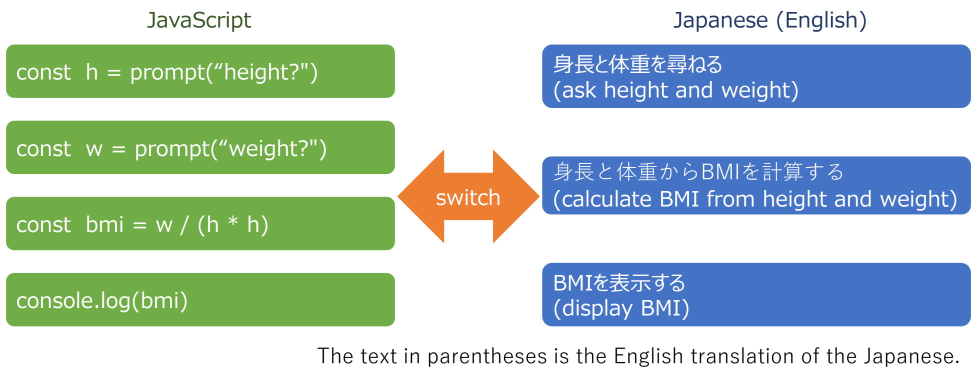

JavaScript
const h = prompt(“height?")
const w = prompt(“weight?")
const bmi = w / (h * h)
console.log(bmi)
switch
Japanese (English)
身長と体重を尋ねる
(ask height and weight)
身長と体重からBMIを計算する
(calculate BMI from height and weight)
BMIを表示する
(display BMI)
The text in parentheses is the English translation of the Japanese.

Figure 1: Conceptual drawing of language switching

2.2 Support for various languages by registering language assets

In the visual programming environment, the limited choice of programming languages is often a problem. For example, Blockly[17] supports JavaScript, Python, PHP, Lua, Dart. This problem is solved in our system by providing a function for language asset registration. Language assets are data sets for each programming language. The user (teacher) registers language assets for use in education such as "C# assets", "Japanese assets", and "JavaScript assets". Language assets have language names, blocks, code written in JavaScript to execute blocks, information for display such as icons, and so on.

By allowing the teacher to register multiple language assets in this way, the learner is able to practice programming in various languages, and thus this learning support system can support the understanding of a wide range of programming languages.

2.3 Support for understanding program behavior by highlighting the currently executing block

For beginners in programming, it is difficult to understand how a program works just by looking at the source code[7][18]. To solve this problem, this learning support system provides a function for highlighting the currently executing block, as well as a function for highlighting variables whose values have been changed.

The running block highlighting function changes the color of the running block to highlight it. By using it together with a function for changing the execution speed, the user is able to monitor the processing flow.

In addition, a function for highlighting changing variable values allows the user to check changes in variable values during program execution.

2.4 Support for individual student guidance by showing students' learning status to teachers

In conventional research and with conventional tools, it is difficult for teachers to grasp the programming thought processes of individual learners. In addition, it is not possible for teachers to confirm the frequency of use of the display language switching function and its effectiveness. In order to solve these issues, our system collects application operation logs for each learner, visualizes them, and presents them to teachers.

This function enables teachers to grasp the programming process of each learner and give individual guidance. In addition, the system makes it possible to analyze the relationship between the learning effect and (i) the frequency with which learners switch display languages, (ii) how long each language is displayed for, and (iii) how long is spent learning. In this way, the system can contribute to the study of effective learning methods.

3 Functions of the Learning Support System

This system features a language asset registration function, a visual programming function for mouse-operated programming and execution, and a learning status display function.

The visual programming function allows learners to create programs without using a keyboard. In addition, programs can be executed in a form that is easy to understand visually. With the visual programming function, the user can use multiple programming languages at the same time and execute code while being able to switch the display language at any time.

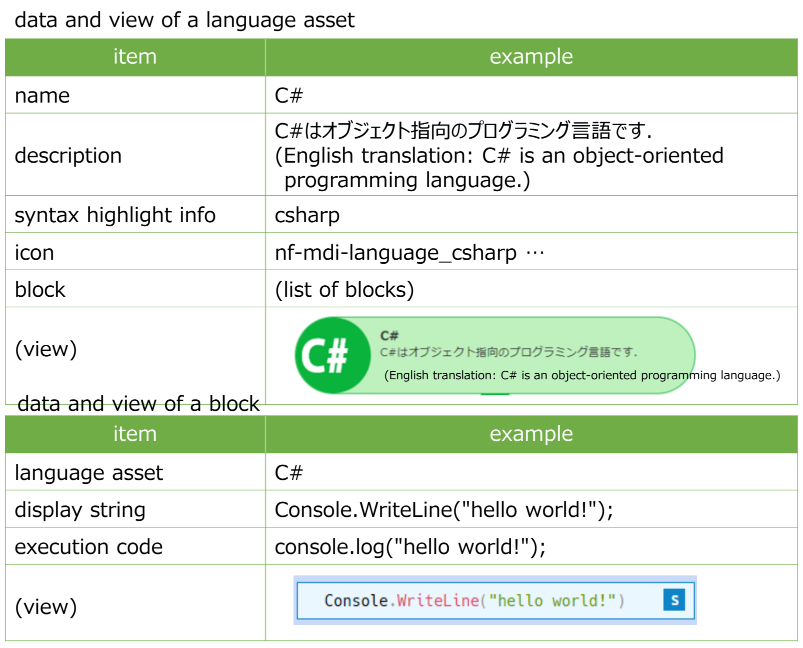
Teachers can register language assets to be used in visual programming and make them available to learners. Language assets registered by teachers and projects created by learners can be saved and shared on the Web. In addition, the teacher can browse the learning status of their students. The details of the language asset registration function for teachers are described in 3.1, the visual programming function for learners is described in 3.2, and the learning status display function is described in 3.3.

3.1 Language asset registration

The language asset registration function is a function for registering information and code blocks related to languages used in visual programming.

As shown in Figure 2, language names and descriptions, display information such as icons, and a list of blocks are all registered as language assets. For example, if a teacher creates a "C#" language asset and registers a list of C# blocks, learners will be able to use C# in the system.

The blocks registered in a language asset are the objects that the learner manipulates by dragging and dropping during visual programming. It consists of display text and executable code.



data and view of a language asset

item	example
name	C#
description	C#はオブジェクト指向のプログラミング言語です. (English translation: C# is an object-oriented programming language.)
syntax highlight info	csharp
icon	nf-mdi-language_csharp …
block	(list of blocks)
(view)	C# C#はオブジェクト指向のプログラミング言語です. (English translation: C# is an object-oriented programming language.)

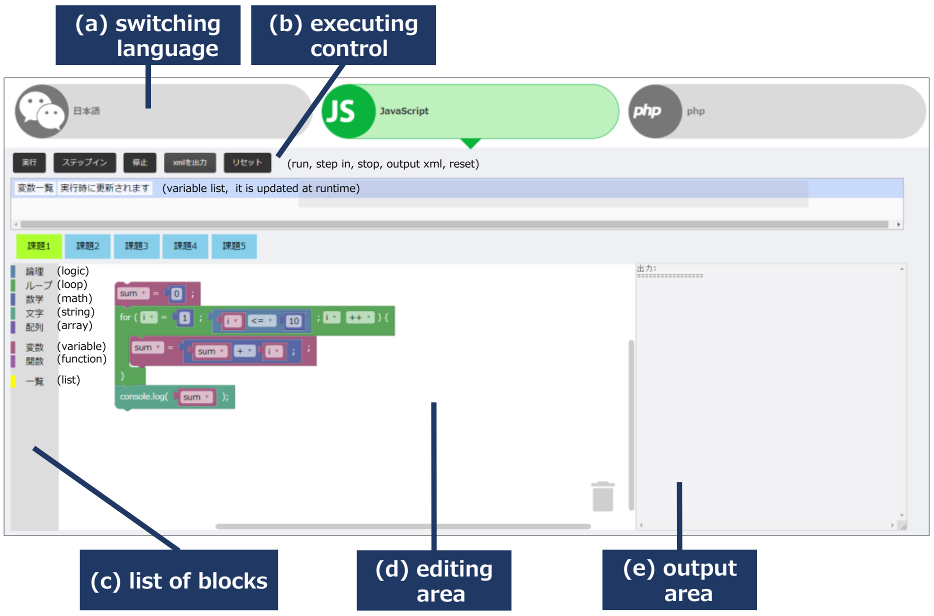
data and view of a block

item	example
language asset	C#
display string	Console.WriteLine("hello world!");
execution code	console.log("hello world!");
(view)	Console.WriteLine("hello world!")

Figure 2: Examples of Language Assets and Blocks

3.2 Visual programming function

This is a function for programming by dragging and dropping blocks. There is also a function for switching to another language, and a function for executing the program graphically in a visually easy-to-understand form. Figure 3 shows an on-screen example of the visual programming function.



```
sum = 0 ;
for ( i = 1 ; i <= 10 ; i ++ ) {
  sum = sum + i ;
}
console.log( sum );
```

The text in parentheses is the English translation of the Japanese on the screen.

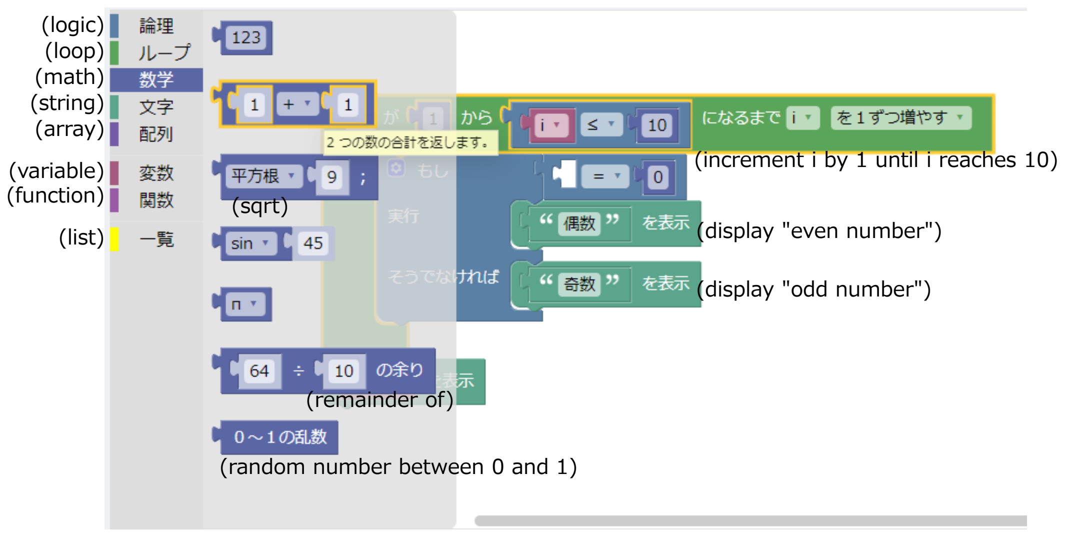
Figure 3: Example screenshot of the learning support system

3.2.1 *Function for programming by drag and drop*

On the left side of the screen, the list of block-categories registered in the currently selected language asset is displayed ((c) in Figure 3). There are seven categories: logic, loops, numbers, strings, arrays, variables, and functions. When the user selects a block category, a list of blocks in that category is displayed, as shown in Figure 4. The user can create a program by dragging and dropping these blocks into the editing area on the right side of the screen ((d) in Figure 3) with the mouse.

The blocks can be changed to another language by using the language switching function described later in this paper. The variables used in the blocks dragged into the editing area are displayed as a variable list at the top of the editing area.

The created program can be deleted by double-clicking the trash can icon displayed at the bottom right of the editing area.

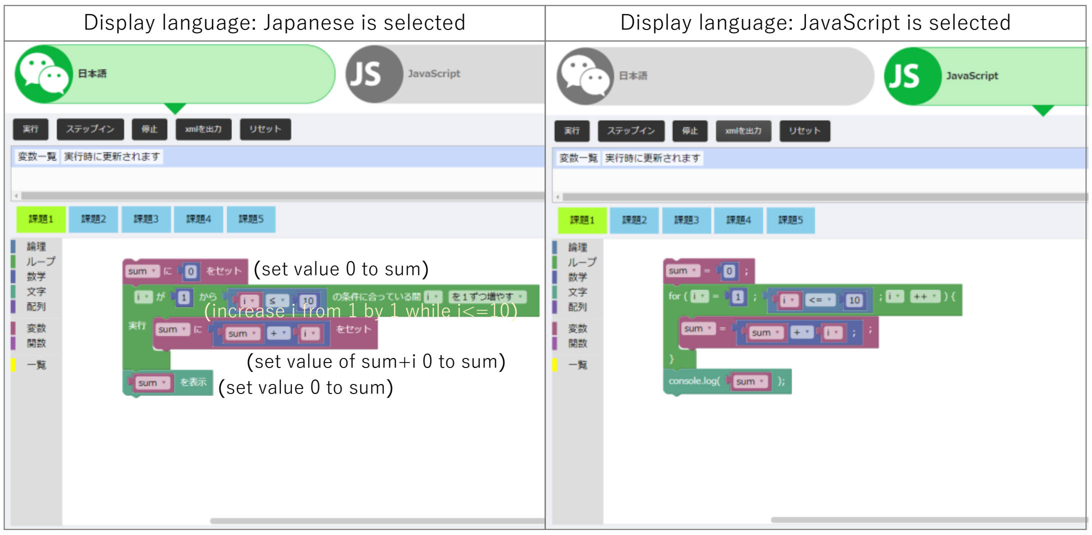


The text in parentheses is the English translation of the Japanese on the screen.

Figure 4: Examples of Block Categories and Blocks

3.2.2 *Function for switching display language*

The user can switch the currently selected language by clicking the language switching buttons ((a) in Figure 3) displayed at the top of the screen. Buttons for all available language assets appear at the top of the screen. By clicking these buttons, the user can switch the display language of the blocks in the block list area and the editing area as shown in Figure 5. At this time, the structure of the program created by the learner in the editing area is retained, and only the display of the block text is switched.



```
sum = 0 ;
for ( i = 1 ; i <= 10 ; i ++ ) {
  sum = sum + i ;
}
console.log( sum );
```

Figure 5: Example of language switching

3.2.3 *Execution function*

The execution function graphically executes a program created by a learner. The program can be executed by clicking the button in the execution control area (Figure 3 (b)). The execution function has the following sub-functions.

a) Step execution: This sub-function supports better understanding of the behavior of a program by executing the program step by step. When the user clicks the step-in button at the top, one block is executed, and when the user presses it continuously, the next blocks are executed consecutively.
b) Change execution speed: The execution speed change sub-function is used to execute a program at a speed that is easy to follow. The speed can be changed by changing the value of the execution delay in the execution control area. For example, if this is set to 1000, the system waits 1000ms after each block is executed.
c) Highlight currently executing block: This is a function that emphasizes the currently running block as shown in Figure 6. It is possible to visually check which block is currently being executed even for processing whose flow is difficult to understand, such as that involving repetition and conditional branching.
d) Variable display: Whenever the value of a variable is changed, the value is displayed as shown in Figure 6. By using this function, it is possible to constantly check the value of any variable and understand how the values of variables change as the program is executed.

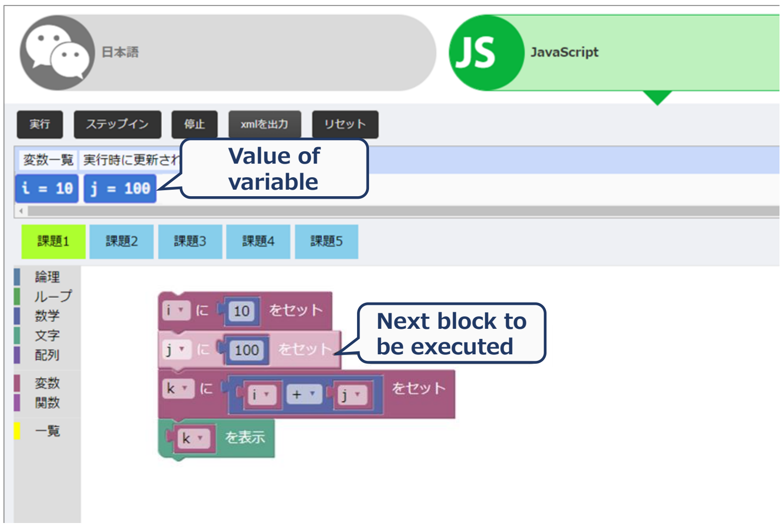


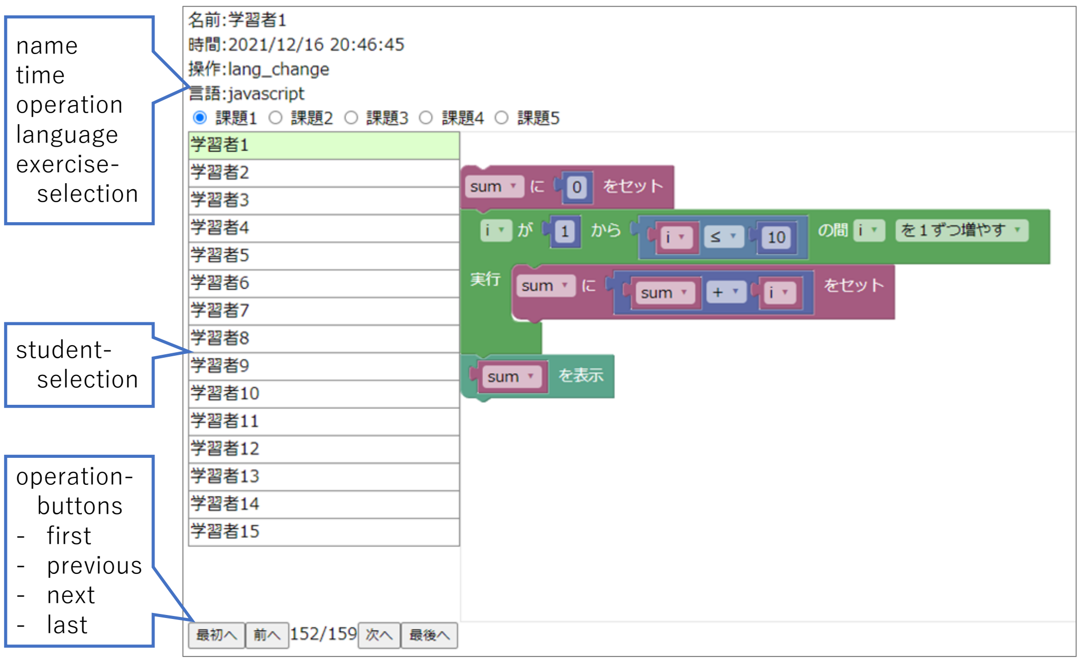
Figure 6: Example of highlighting a currently executing block and displaying variable values

3.3 Learning status display function

This is a function that records the problem-solving process that a learner uses in an exercise-style lesson and provides it as feedback to the teacher.

This function collects logs of learner operations (addition / deletion / movement of blocks, addition of variables, switching of languages) and saves them in a database on the server. This function enables teachers to understand the programming process followed by each learner and to teach them individually. In addition, it becomes possible to analyze the relationship between the frequency with which display languages are switched and the learning effect, or the relationship between the display time for each language and the learning effect, each of which can contribute to the understanding of effective learning methods.

Figure 7 shows the screen for instructors. When the learner's name and assignment number are selected on the teacher's screen, the status of the learner's editing area is displayed on the right of the screen. Teachers can check each learner's situation one at a time by pressing the buttons at the bottom of the screen. The learner's name, operation time, operation type, and the language displayed when the operation was performed are displayed at the top.



名前:学習者1
時間:2021/12/16 20:46:45
操作:lang_change
言語:javascript
[✓ 課題1] 課題2 課題3 課題4 課題5

学習者1
学習者2
学習者3
学習者4
学習者5
学習者6
学習者7
学習者8
学習者9
学習者10
学習者11
学習者12
学習者13
学習者14
学習者15

```
sum に 0 をセット
i が 1 から i ≤ 10 の間 i を1ずつ増やす
実行 sum に sum + i をセット
sum を表示
```

最初へ 前へ 152/159 次へ 最後へ

Figure 7: Example screen showing the learning status display function in use

4 Implementation

4.1 System configuration

This learning support system was developed as a Web application with few restrictions on the usage environment. Node.js, Web application framework Express, and its template engine EJS were used for server-side development.

HTML, JavaScript, and CSS were used for client-side development. The server side consists of database control, and the client side consists of Blockly[17] for block management and JS-Interpreter[1] as the execution environment of the code generated by Blockly.

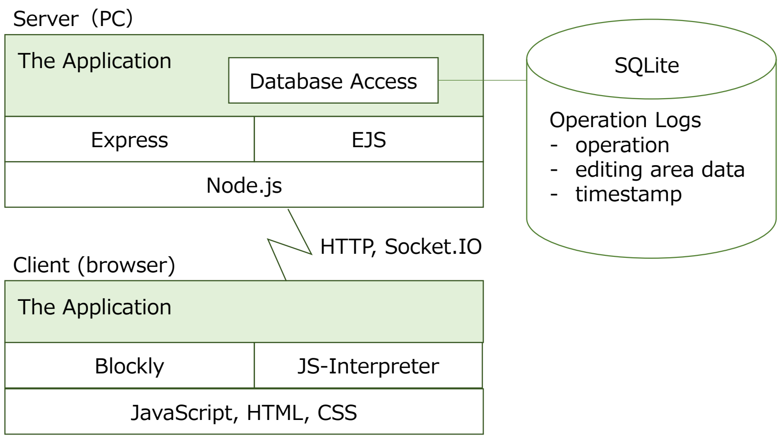


Figure 8: Configuration of the learning support system

4.2 Data model of operation logs

This application uses SQLite as a database management system. The learner ID, exercise ID, task number, operation, displayed language, editing area content, and time stamp are recorded in the application's operation log for each operation.

The list of operations that are recorded is as follows.

(1) create: an operation to add a new block to the workspace;

(2) delete: an operation to delete a block from the workspace;

(3) move: an operation to move a block within the workspace;

(4) var_create: an operation to create a new variable;

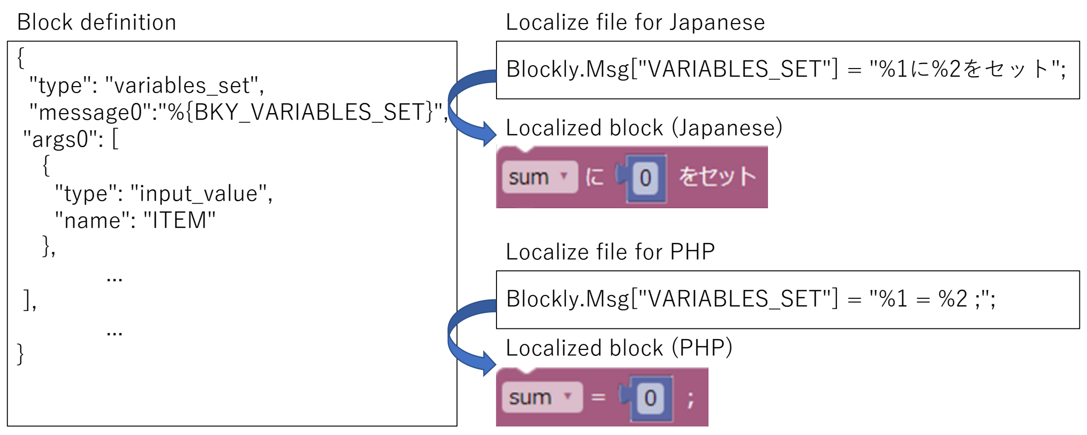
(5) lang_change: an operation to switch the displayed language.

4.3 Implementation of the function for switching the display language

4.3.1 Function for switching display language

Blockly supports block definitions that can be localized to the user's language. The language switching function uses this to implement display of code blocks in either natural or programming language.

Figure 9 shows a definition example for displaying an assignment block in Japanese and PHP. In Blockly, as shown on the left side of Fig. 9, the block definition is described in JSON format (a set of key / value pairs). The value of the key "message0" is displayed in the block. By describing this value as "%{BKY _…}", the character string described in the localization file is referenced, and its contents are displayed as shown on the right-hand side of Fig. 9. The reference "%1" described in the localization file indicates an input position for numerical values and variables.



```
{
  "type": "variables_set",
  "message0":"%{BKY_VARIABLES_SET}",
 "args0": [
   {
     "type": "input_value",
     "name": "ITEM"
   },
          …
 ],
          …
}
```

```
Blockly.Msg["VARIABLES_SET"] = "%1に%2をセット";
```

```
Blockly.Msg["VARIABLES_SET"] = "%1 = %2 ;";
```

Figure 9: Definition example for displaying the assignment block in Japanese and PHP

4.3.2 Code execution function

JS-Interpreter is used to execute the program in this application. JS-Interpreter is a sandboxed open source JavaScript interpreter written in JavaScript. You can execute arbitrary JavaScript code line by line. Also, if multiple instances of JS-Interpreter are created, each piece of code can be executed in multiple threads at the same time. These are completely separated from the JavaScript environment, and it is therefore possible to execute JavaScript code safely.

The program built in the editing area is converted into a JavaScript statement and executed line by line using the step function of JS-Interpreter. In this system, every time the step-in button is clicked, the step function is called and the block is executed line by line. Also, when the execute button is clicked, the block is executed line by line with the user-specified time interval.

4.3.3 Variable value display function

Since JS-Interpreter is separated from the JavaScript execution environment of the browser, variables cannot be directly referenced from outside it. This system uses the JS-Interpreter API to obtain the values of variables. In JS-Interpreter, the API can be defined and called in a way similar to JavaScript function definitions. In this system, we have used the API to retrieve the correspondence between variables and their values. The API is also used to rewrite the value of a variable each time it is changed.

4.4 Implementation of learning status display function

In the learning status display function, when the learner takes any action, including adding / deleting / moving blocks, creating variables, or switching the display language, the information is sent to the server and saved in the database.

These events are detected and operation logs are collected through Blockly's event listener. The collected information is sent to the server using the Node.js library Socket.io and stored in the database.

On the teacher's screen, when the application window is opened, the learning behavior log collected up to that time is displayed. Once the teacher screen is opened, the latest learning activity log is retrieved from the server and updated every 20 seconds thereafter.

5 Evaluation

Two trial experiments were conducted to evaluate the usefulness and usability of this learning support system. The first trial experiment was conducted when the prototype of this system was completed. The second trial experiment was conducted using a system with expanded functions and improved usability developed on the basis of the results from the first experiment. The results of each experiment are described below.

5.1 First experiment

A trial experiment was conducted to evaluate the usefulness and usability of this learning support system. The subjects of the experiment were six undergraduate students. The subjects were given an overview of the system and an explanation of its operation, and were asked to try it. After that, we conducted a questionnaire survey containing the following questions about the usefulness of the system.

a. Did you achieve a better understanding of the language than you had before using this tool?

b. Did the language switching function help you to learn the syntax and function / property names?

c. Did the function for changing the code execution speed help you understand the flow of the program?

d. Did the variable display function help you to get an idea of the relationship between the execution result and the code?

e. What features of the system helped you understand the program?

In the experiment, we used "Japanese assets," "JavaScript assets," and "Nadeshiko assets," and instructed the participants to switch languages as appropriate while they were studying.

Figure 10 shows the results of the questionnaire regarding the usefulness of this learning support system. These items were answered by selecting one of 5 options from 1 (useless) to 5 (useful).

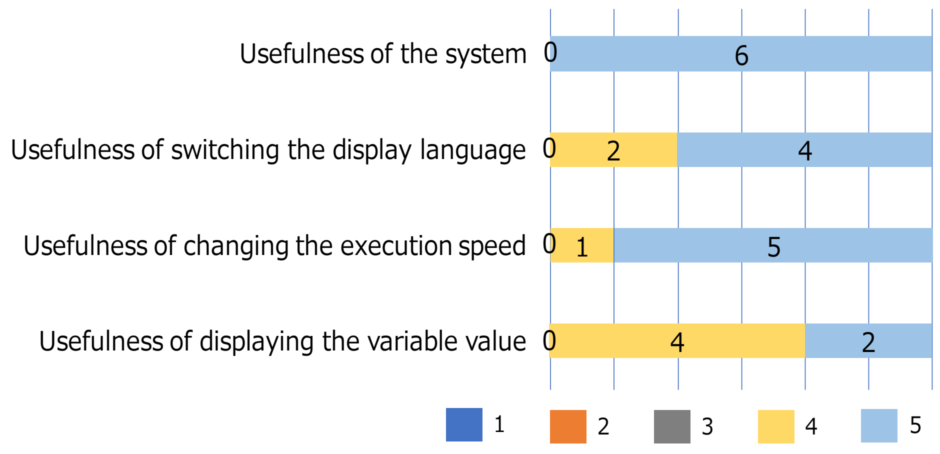


Figure 10: Questionnaire results on usefulness

All 6 subjects answered that they achieved a better understanding of the programming language than before using the system. Also, regarding the usefulness of each function, all the answers were 4 or higher on the 5-point scale. From these responses, it can be said that this system is effective for learning.

However, the variable display function was rated lower than the other functions. It needs to be improved so that the values of variables can be confirmed more easily.

In addition, responses about the features of the system that most aided a better understanding of the program mentioned language switching, execution speed change, execution location display, and the variable value display. Since these functions are the characteristic functions of this system, it was confirmed that the system can support programming learning as effectively as other systems.

5.2 Second evaluation experiment

The subjects of the second trial experiment were eleven undergraduate students and one faculty member, for a total of twelve people. In the experiment, a brief explanation of the system was given to the subjects, and some task exercises using the system were assigned. We also conducted a simple written test about PHP and questionnaire surveys before and after the exercise.

In this experiment, three types of localized files, "Japanese", "JavaScript", and "PHP", were registered. The participant learns about PHP while switching the display language and proceeds with the task.

A questionnaire survey was conducted in advance on the subjects' programming experience and the extent their knowledge about the particular programming language. The survey offered five options: 1 (can't do anything at all) to 3 (can create a program of several tens of lines) to 5 (have experience in developing applications). The results are shown in Fig. 11. The subjects' asserted knowledge of programming in general was as high as 5 at maximum, 3 at minimum, 4 at median, and 3 at mode, and the extent of their asserted knowledge about PHP in particular was as low as 3 at maximum, 1 at minimum, 2 at median, and 2 at mode..

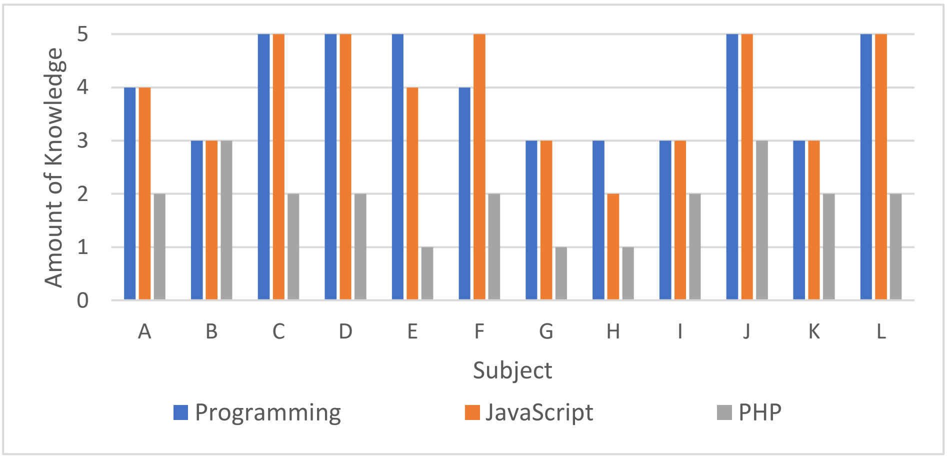


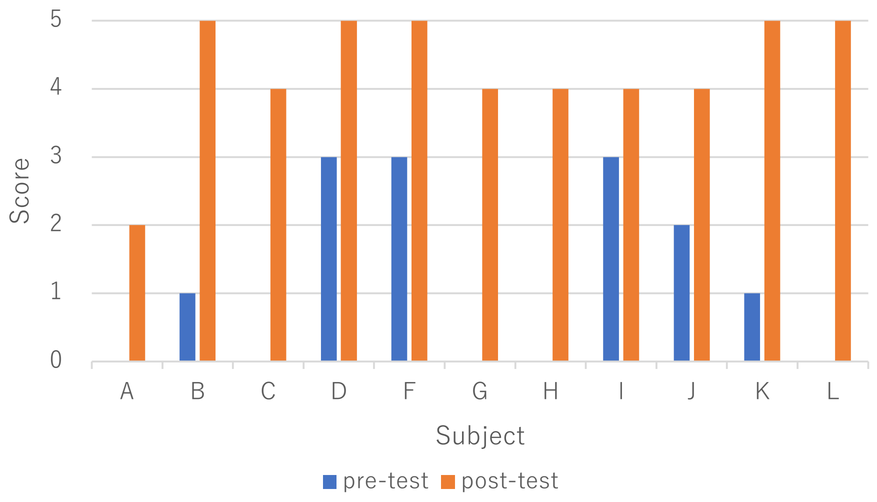
Figure 11: Knowledge of Programming

5.2.1 Learning effect

The results of the pre-experiment and post-experiment tests of learning using this system were compared, and the learning effect was evaluated from the results.

The subjects answered a 5-point scale test about PHP before and after the exercise. Figure 12 shows the test results excluding a subject who had a perfect score in the pre-test and post-test. Between the pre-experiment and post-experiment tests, the average score increased by 3.09 points. A paired t-test gave a t-value of 7.88 which is greater than 3.169 (10 degrees of freedom, 1% significance level), so the null hypothesis "there is no difference in scores between the pre-test and post-test" is rejected. In the results of the post-experiment test, there were many syntactical mistakes in the description of "$" representing PHP variables and ";" at the end of statements.

From these results, it was confirmed that the use of this system to teach a programming language can effectively aid learner understanding of how to use functions and about the flow of a program, but the effect is limited as regards the reinforcement of the details of syntax. It is thought that this is because this system automatically adds the variable marker "$" and the statement ending ";".



	pre-test	post-test	increase amount
average	1.28	4.27	3.09
median	1	4	4
invariant variance	1.75	0.81	1.69

standard error of difference: 0.39, t-value: 7.88

Figure 12: Results of pre-experiment and post-experiment tests

5.2.2 *Questionnaire survey*

After the experiment, we conducted a questionnaire survey on the good bad points of using this system.

The reported good points were as follows.

(a) I have learned JavaScript, and by switching languages, I can understand the corresponding functions at a glance.

(b) It was easy to understand the behavior of the program because the program could be constructed using Japanese.

(c) By switching the programming language, the user can understand how to write in multiple languages, which reduces the time and effort required to learn.

(d) By using blocks, users can learn while playing intuitively as if playing a game.

(e) I feel that it is very useful for learning PHP.

(f) You can write a program without learning detailed syntax.

(g) The step execution function allows the user to thoroughly understand the behavior of the program.

The reported bad points were as follows.

(a) I don't understand how to use the ‘else’ block.

(b) Two semicolons are displayed in the assignment statement.

(c) The system freezes when an infinite loop occurs.

(d) There is no feedback when I send the answer.

(e) Block manipulation is troublesome.

From these results, it was confirmed that this system is useful for learning a new programming language. In addition, since functions that do not feature in existing applications such as Scratch are being evaluated, the system is considered to be useful as a tool for learning new programming languages.

5.3 Relationship between learning behavior and learning effect

The relationship between learning behavior and learning effect was analyzed from the collected operation logs and the amount of increase in test points. For the analysis, we used the data relating to 11 subjects excluding those who had a perfect score in the preliminary test. Table 1 shows a summary of the test results and operation logs.

Figure 13 shows a scatter plot of the number of language switches and the amount of increase between the test scores before and after the exercise. Its correlation coefficient is -0.01 and there is no correlation between them. This is thought to be because the timing of language switching differs depending on learner preference, as follows.

・The learner wants to learn while diligently switching to other languages

・The learner wants to learn only in a specific language without switching to other languages;

・The learner does not switch languages while creating a program. He/she switches languages to make checks after the program is completed.

Table 1: Test results and summary of operation logs

subject	pre-test (points)	post-test (points)	increase amount (points)	number of operations (times)				language switching count (times)
				sum	while displaying Japanese	while dislaying JavaScript	while displaying PHP	
A	0	2	2	289	30	28	231	24
B	1	5	4	382	59	67	256	30
C	0	4	4	367	204	17	146	41
D	3	5	2	762	622	124	16	45
F	3	5	2	335	104	0	231	2
G	0	4	4	515	32	194	289	6
H	0	4	4	338	221	34	83	5
I	3	4	1	355	355	0	0	1
J	2	4	2	197	0	0	197	0
K	1	5	4	291	0	0	291	0
L	0	5	5	239	0	1	238	1

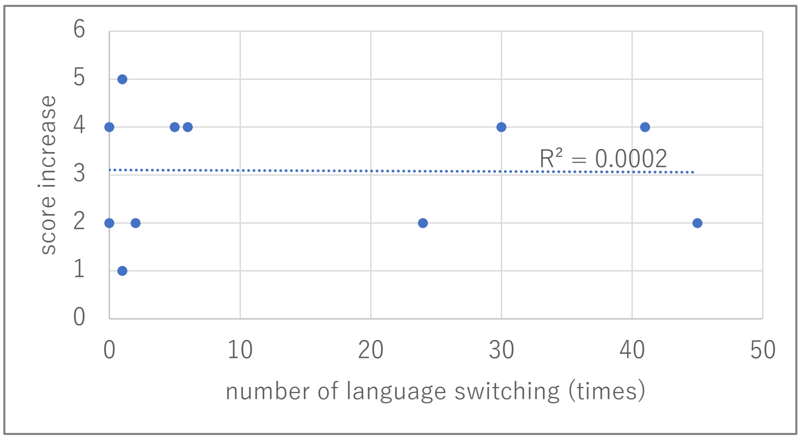


Figure 13: Scatter plot of the number of language switches and score increase

Figure 14 shows a scatter plot of the number of operations made while displaying each language and the amount of increase in the test score. The three graphs on the left show the rate of operations and the amount of increase in test score, and the three graphs on the right show the number of operations and the amount of increase in test score. The correlation coefficient between the rate of operations and the amount of increase in test score was -0.45 for Japanese, 0.24 for JavaScript, and 0.37 for PHP. The correlation coefficient between the number of operations and the amount of increase in test score was -0.43 for Japanese, 0.15 for JavaScript, and 0.51 for PHP. A negative correlation was obtained for Japanese display and a positive correlation was obtained for PHP display. No correlation was found for JavaScript display.

The number of subjects is small, and therefore the results may not be definitive, but the results do show that the more times PHP is displayed and manipulated, the higher the learning effect, and the more times Japanese is displayed and manipulated, the lower the learning effect. It was confirmed that it was possible to analyze the relationship between the learning effect and how much learning was done while displaying each language. By applying this method to many learners and accumulating the log data, it can be used for studying effective learning methods.

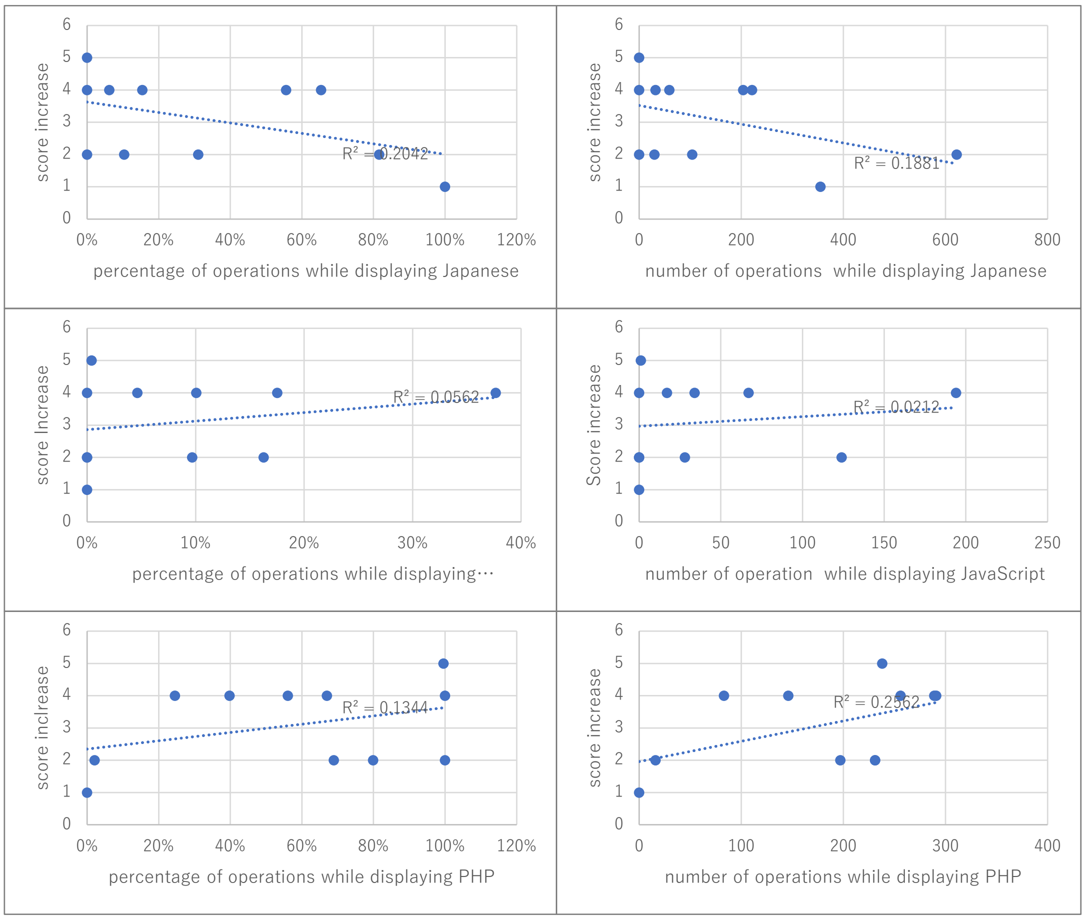


Figure 14: Scatter plot of number of operations and test score increase

5.4 Future work

Future issues are as follows.

(1) Expansion of supported language functions: This system supports concepts common to many programming languages, such as three basic structures, functions, basic data types, and arrays. However, it does not support language functions such as variable scope, structs, and objects. These are important concepts for learning programming languages, and their support is a topic for future work.

(2) Individually optimized education using system operation logs: In this research, we developed a function to visualize learning status using the system operation logs. By accumulating and analyzing a large amount of collected operation logs, it is possible to extract the programming ideas and styles (top-down or bottom-up) of each learner. Also, if there is an error in the programming direction (such as a learner failing to implement a loop for a problem that requires repetition), hints can be given at appropriate times. The realization of these features is a topic for future work.

(3) The number of subjects is small, and therefore the results may not be definitive. Evaluation by a larger number of subjects is a future work. Evaluation by dividing the subjects into an intervention group and a control group is also a future work.

6 Conclusion

In this research, we have developed a programming learning support application that can switch between display languages for the purpose of assisting the understanding of code and program operation.

We have implemented the following system functions: a function to switch the language in which blocks of code are displayed; a visual programming function that allows the construction of programs using drag-and-drop operations; a function to highlight currently running blocks of code and relevant variables. In addition, a learning status display function has been implemented for the use of teachers.

In order to evaluate the usefulness of this application, we conducted a questionnaire survey of users and evaluated the effected change in understanding of programming technology as a result of using the system, and the system's ease of use. The results of the experiment confirmed that this tool aids user understanding of code and its behavior when learning a new programming language. We also confirmed that it is possible to analyze the relationship between the learning effect and how much learning was done while displaying each language.

Future tasks include further evaluation of learning outcomes for learners using this system, evaluation by a larger number of subjects, and examination of the more effective use of log data.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number JP20K03078.

References

[1] Mobile marketing data labo., January 2020 Awareness Survey on Programming Education for Elementary and Junior High School Students, https://mmdlabo.jp/investigation/detail_1830.html (in Japanese) (accessed 31 Dec. 2021).

[2] Mitchel Resnick, et al., Scratch: programming for all, Communications of the ACM, Vol.52, No.11, pp.60-67, 2009.

[3] John Maloney, Mitchel Resnick, Natalie Rusk, Brain Silverman and Evelyn Eastmon, The Scratch Programming Language and Environment, ACM Transactions on Computing Education, Vol.10, No.4, Article 16, 2020.

[4] Swift Playgrounds – Apple, https://www.apple.com/swift/playgrounds/ (accessed 31 Dec. 2021).

[5] Japanese Programming Language Nadeshiko, https://nadesi.com/ (in Japanese) (accessed 31 Dec. 2021).

[6] Mineaki SAKATOKU, Japanese Programming Language Nadeshiko, Computer Software, 2011, Volume 28, Issue 4, pp.23-28, 2011 (in Japanese).

[7] D. Bau, J. Gray, C. Kelleher, J. Sheldon, F. Turbak, Learnable programming: blocks and beyond, Communications of the ACM, June 2017, pp. 72-80, 2017.

[8] M. Armoni, O. Meerbaum-Salant, M. Ben-Ari, From Scratch to “real” programming, ACM Transaction on Computing Education, Vol.14, No.4, Article No. 25, 2015.

[9] Y. Matsuzawa, Y. Tanaka, S. Sakai "Measuring an impact of block-based language in introductory programming". In: Brinda T., Mavengere N., Haukijarvi I., Lewin C., Passey D. (eds) Stakeholders and Information Technology in Education. SaITE 2016. IFIP Advances in Information and Communication Technology, vol 493. pp.16-25, Springer, Cham, 2016.

[10] T. W. Price, T. Barnes, Comparing textual and block interfaces in a novice programming environment, Proceedings of the eleventh annual International Conference on International Computing Education Research, pp.91-99, 2015.

[11] Top 10 In-Demand programming languages to learn in 2020, https://towardsdatascience.com/top-10-in-demand-programming-languages-to-learn-in-2020-4462eb7d8d3e (accessed 31 Dec. 2021).

[12] Programming language rankings: JavaScript still rules, Python holds off Java, https://www.zdnet.com/article/programming-language-rankings-javascript-still-rules-python-holds-off-java/ (accessed 31 Dec. 2021).

[13] Hussein Alrubaye, Stephanie Ludi, Mohamed Wiem Mkaouer, Comparison of block-based and hybrid-based environments in transferring programming skills to text-based environments, Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering, pp.100–109, 2019.

[14] R. Benjamin Shapiro, Matthew Ahrens, Beyond blocks: syntax and semantics, Communications of the ACM, Vol.59 Issue 5, pp.39–41, 2016.

[15] Kujira Hikou Dukue, About Japanese Programming Language "Nadeshiko", IPSJ magazine, Vol.62, No.5, pp.e26-e42, 2021 (in Japanese).

[16] Naoya Kamita, Tetsuo Tanaka, Prototyping and Evaluation of Programming Learning Support System with Function for Switching Display Language, 10th International Congress on Advanced Applied Informatics (IIAI AAI 2021) Proc., pp.129-134, 2021.

[17] Blockly Reference, https://developers.google.com/blockly/reference/overview (accessed 31 Dec. 2021).

[18] Austin Z. Henley, Julian Ball, Benjamin Klein, Aiden Rutter, Dylan Lee, An inquisitive code editor for addressing novice programmers' misconceptions of program behavior, Proceedings of the 43rd International Conference on Software Engineering: Joint Track on Software Engineering Education and Training, pp 165–170, 2021.

[19] JS-Interpreter Documentation, https://neil.fraser.name/software/JS-Interpreter/docs.html (accessed 31 Dec. 2021).