Proposal and Evaluation of Intermediate Content for the Transition from Visual to Text-based Programming Languages

Katsuyuki Umezawa
Shonan Institute of Technology, Japan
umezawa@info.shonan-it.ac.jp

Kouta Ishida
Arrows Systems Co. Ltd., Japan
ishida-k@arrows-systems.co.jp

Makoto Nakazawa
Junior College of Aizu, Japan
nakazawa@jc.u-aizu.ac.jp

Shigeichi Hirasawa
Waseda University, Japan
hira@waseda.jp

Abstract

Beginners learning to program learn visual-based programming languages, such as Scratch, whereas experts use text-based programming languages, such as C and Java. However, no seamless transition from visual to text-based programming languages has been established. In this paper, a transition method was established between both language types. In particular, emphasis was placed on the features that an intermediate language between visual and text-based programming languages should have. Additionally, learning with the proposed intermediate content enhanced the subsequent comprehension of text-based languages. The proposed intermediate content was evaluated using questionnaires to ensure that it had intermediate characteristics between both language types.

1. Introduction

Visual-based programming languages (referred to as visual-based languages) have recently been used to introduce beginners to programming. Subsequently, they are introduced to text-based programming languages (referred to as text-based languages), such as C and Java languages. However, no seamless migration method has been established or recorded.

A research project aimed at establishing a transition method from visual- to text-based languages has been started. Specifically, we study and prototype educational content (called an intermediate language) that can be learned using visual- and text-based languages, thereby bridging the differences between the two languages. Furthermore, we want to evaluate the result and the learning state during learning through the empirical experiment. In this way, we would like to evaluate the effectiveness of the intermediate language and complete educational content that will be useful for future primary and secondary programming education.

Several studies have been conducted on the evaluation of intermediate languages to fill the gap between visual- and text-based languages. However, these studies only focused on the results of the learning effects, such as post-learning questionnaires and grades, and only evaluated whether learners understood. However, these evaluation methods cannot accurately measure the effect of intermediate languages. In this paper, in addition to the conventional evaluation method after learning, biological information, such as brain waves, eye-tracking information, heartbeat, and facial expressions during learning and the learning state, were measured. Then, we analyzed and evaluated whether the intermediate language plays an intermediate role between visual- and text-based languages and contributes to a smooth transition. Once this research is established, it is expected that beginners in programming languages can start learning with visual-based languages and seamlessly and spontaneously transition to learning text-based languages.

Herein, intermediate contents were first proposed as part of the abovementioned research project. It was shown through empirical experiments that the degree of understanding the text-based language improved using the intermediate contents between both language learning methods. Additionally, the proposed intermediate content was evaluated using

URI: https://hdl.handle.net/10125/102639
978-0-9981331-6-4

questionnaires to ensure that its characteristics are positioned between the visual- and text-based languages.

2. Previous Work

2.1. About Visual-based Languages

Visual-based languages fall into two major categories: block-based imperative languages and flow-based functional languages.

Mason and Dave (2017) conducted hundreds of experiments to create simple problems designed to be similar in the block-based and flow-based languages and conducted an empirical study to evaluate the relative benefits of both categories.

Robinson (2016) studied the transition from a block-based language (Scratch) to a text-based language. One fascinating feature of Scratch is that learners can avoid syntax errors and easily understand and use them. Scratch teaches beginners logical thinking rather than writing programs as it places a strong emphasis on understanding programming logic.

2.2. Comparative Study of Visual-based and Text-based Languages

Mladenović, Boljat, and Žanko (2018) surveyed student misunderstandings on loops in 207 elementary school students, a basic concept of programming. The students learned three programming languages: block-based language (Scratch) and text-based languages (Logo and Python). They observed that block-based languages minimized misunderstandings on loops. This difference became more obvious as the tasks became more complex, such as nested loops. They defended the necessity of using a visual-based language for novice programmers as it prevents syntax errors. However, there is no mention of bridging the gap between the languages.

Navarro-Prieto, and Cañas (2001) conducted experiments to explain why visual-based languages are easier to understand than text-based languages in terms of image processing psychology. They tested the hypothesis that visual-based languages build mental representations of data flow relationships faster than text-based languages as image processing speeds up access to semantic information. Results showed that programmers using text-based languages first access the mental representation of the control flow before that of the data flow.

Xu, Ritzhaupt, Tian, and Umapathy (2019) examined existing academic databases and observed the overall impact of block- and text-based programming environments on cognitive- and emotional-learning outcomes of students. However, they were unable to show the statistical advantages of using block-based language and its efficiency for novice programmers. However, they stated the importance of further study of hybrid languages.

2.3. Proposal of Hybrid-based Language

Daskalov, Pashev, and Gaftandzhieva (2021) proposed an environment for beginners to use a hybrid language of text- and visual-based languages. It is a hybrid-based environment of flow-based visual and text-based languages instead of block-based languages. They claimed it is suitable for training novice programmers.

Weintrop (2017) compared text-, visual-, and hybrid-based languages and concluded that while hybrid-based languages showed characteristics of both language types, they outperformed block- and text-based languages in certain dimensions.

Weintrop (2015) used PencilCode to compare the impact of migration skills using block-, hybrid-, and text-based languages. Three groups of learners studied in each language for five weeks, and after the sixth week, all groups switched to text-based language using Java. Learners using block-based languages had a better understanding of loops and variables than those of text-based languages. However, the drawback of block-based languages is the difficulty in building large and complex programs. The hybrid-based languages did not obtain a fair learning comparison as they were employed by switching between the text- and block-based languages.

Alrubaye, Ludi, and Mkaouer (2019) evaluated the transition by creating tools that displayed blocks only, text only, or blocks and text side -by -side. They discovered that, on average, the hybrid-based approach improved students ' understanding of basic programming concept, memorization, and ease of migration by more than 30% compared with the block- and text-based learning approaches.

2.4. Studies on the Gap Between Visual-based and Text-based Languages

Tóth, and Lovászová (2021) highlighted a gap between visual- and text-based languages. They observed the transition from a visual-based language (MIT App Inventor 2) to a text-based language (Android Studio) using a Java bridge code generator as a mediator of knowledge transfer.

Weintrop, and Wilensky (2019) experimentally evaluated changes in knowledge transfer between learners who started with visual-based languages and those with text-based languages. No significant differences were observed between the language types. Their study did not evaluate the transition from the visual-based to the text-based languages but compared the knowledge after mastering text-based language skills.

Additionally, this research promotes the use of electroencephalography (EEG) to monitor the learning progress of learners. The EEG information was acquired during a keyboard typing task and showed that the value of β/α increased with the difficulty of the task (Umezawa et al, 2018, Umezawa et al, 2020). In this study, considering the EEG evaluation, a difference was confirmed in the EEG when solving problems in a visual-based programming language (Scratch) and a text-based programming language (C). Specifically, for the visual-based programming language, the value of β/α did not increase with increasingly difficult tasks. This result provides different thought pathways used during the learning process of visual and text-based programming languages (Umezawa et al, 2021). .

3. Proposal of Intermediate Content

As shown in Sections 2.2 and 2.3, there have been many studies comparing visual- and text-based programming languages and hybrid programming languages. Against these, we want to clarify the gap between visual- and text-based languages first, rather than simply developing a hybrid language. Also, some studies want to clarify the gaps mentioned in Section 2.4. However, we would ultimately like to measure biological information, such as changes in brain waves and facial expressions during learning, to clarify invisible gaps.

In our research project, we plan to (1) evaluate the learning effect after learning, (2) evaluate the learning state using biometric data during learning, and (3) finally develop and evaluate a new intermediate language. This paper is related to (1).

In the following section, intermediate contents are proposed with visual- and text-based programming language characteristics.

3.1. Overview

The characteristics of the current visual-based language (pleasant appearance, immediate execution, no grammatical errors, etc.) and those of the text-based language (only characters,; fails to work if one character is wrong, and takes a lot of time for graphical representations) were compared and examined. The results show that the proposed intermediate content is simple (no additional knowledge peculiar to text-based languages required), has quick feedback (execution results can be understood immediately), has no grammatical errors, and has easy-to-understand logic error parts.

3.2. Realization of Intermediate Content

Based on the abovementioned characteristics, learning content that uses JSFiddle (2022) to create music by employing JavaScript was developed. In Figure 1, JSFiddle is a web version of the integrated environment that allows coding in JavaScript and displays execution results on a single screen using only a web browser. Music can easily be created by adding a library (Beeplay) for playing sounds in the JSFiddle. For example, to make the sound of "CDE," it can be coded, as shown in Figure 2.

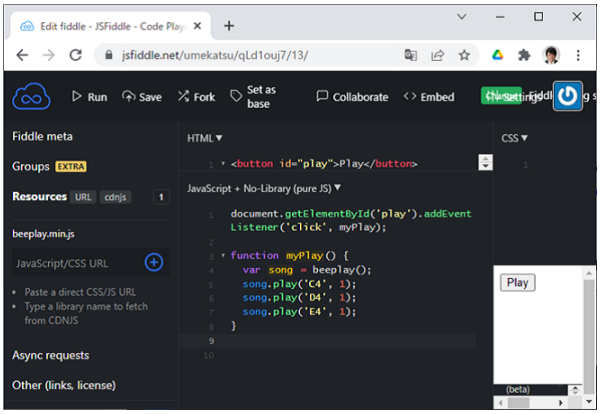

Figure 1. Screen of JSFiddle

A program that creates sounds can be realized only by a method called play. Therefore,

```
function myPlay() {
  var song1 = beeplay();
  song1.play('C4', 1);
  song1.play('D4', 1);
  song1.play('E4', 1);
}
```

Figure 2. A Program that Sounds of CDE

memorizing many words (reserved words) and grammar rules is unnecessary. Upon execution, a sound is heard and the result can be confirmed quickly. Additionally, one can easily judge by listening to the sound at the location of the logic error (the compiler does not output the error because the program is not mistaken) what makes the sound different from the expected sound. Sometimes, a different sound is made during the second repetition. as the music has a repeating structure. This repeating structure can be expressed programmatically as repetitions (for statements) and conditional branches (if statements). In this way, music and programming are compatible. Moreover, creating our favorite music gradually becomes a fun process.

4. Tools Used in the Experiment

In this chapter, the visual-based language, intermediate contents, and text-based language used in the experiment are explained.

4.1. Google Blockly

Google Blockly (2022) is a library provided by Google for creating visual programming. Blockly Games are contents created using Google Blockly to learn visual-based programming that runs on the web. The characteristics of Blockly Games as a programming language are that the execution results are seen immediately, sound can be heard as the execution results, and error is easy to understand. Various contents, such as puzzles, mazes, and music, can be selected on the menu screen.

4.1.1. Blockly Games Puzzles The Blockly game puzzle screen is shown in Figure 3. This puzzle familiarizes participants with visual-based programming. You can easily operate the block by dragging and dropping it with the mouse. For example, connect the image of a cat to the block labeled "Cat" and connect the blocks with the characteristics of the cat to complete the puzzle. You can match the answers by selecting the number of legs from the pull-down menu and pressing the answer button.

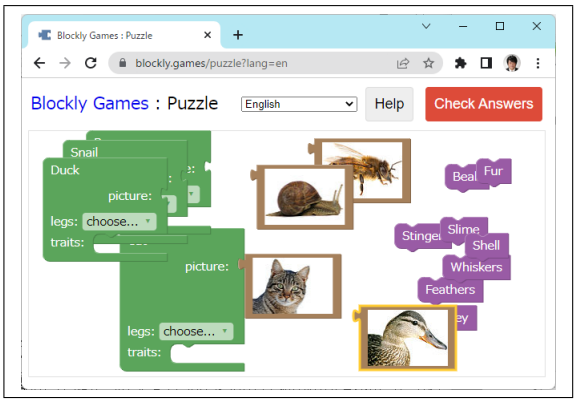

Figure 3. Screen of Blockly Games (Pazzle)

4.1.2. Blockly Games Maze The screen of the Blockly game maze is shown in Figure 4. This content is a humanoid character that goes on the yellow line. The game progresses by dragging and dropping blocks such as "Go straight" to the field on the right side. It was hypothesized that simple iterative processing and conditional branching could be learned in this maze.

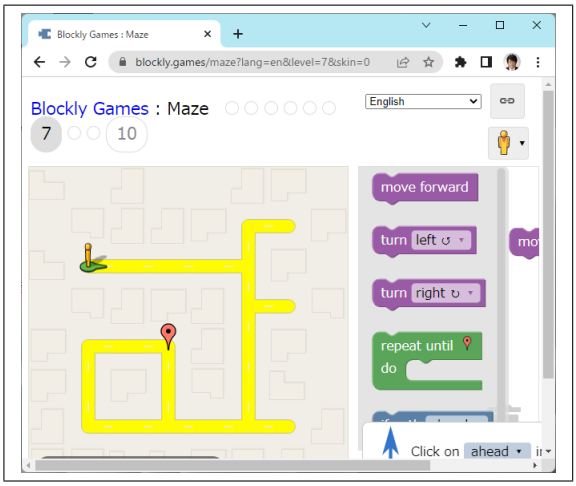

Figure 4. Screen of Blockly Games (Maze)

4.1.3. Blockly Games Music The screen of Blockly game music is shown in Figure 5. This content can be created by dragging and dropping a note block into the field on the right and connecting it to the start block. The game progresses by appropriately changing the note blocks to match the

score on the left side and connecting them.

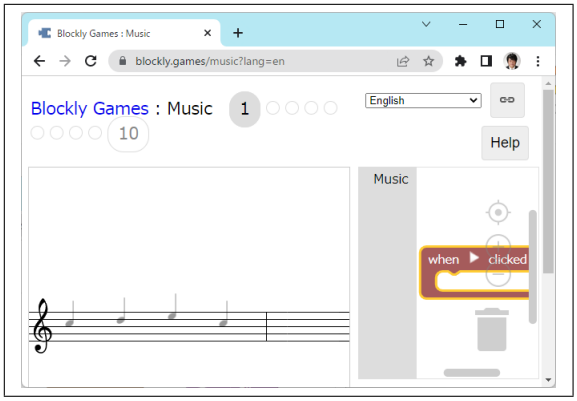

Figure 5. Screen of Blockly Games (Music)

4.2. JSFiddle

As mentioned in Section 3.2, JSFiddle is a web version of the integrated environment that codes JavaScript and displays execution results using only a web browser.

4.3. Python

Python is a text-based programming language. Unlike C and Java programming languages, it has a relatively simple description. Therefore, it was used as the text-based language for this experiment. The purpose of this content is to confirm the learning effect of Blockly Games and JSFiddle mentioned above. The execution environment of Python used Google Colaboratory (2022). The Google Colaboratory is an integrated environment where Python can be written and executed on a browser. Some of the Python problems with Google Colaboratory are shown in Figure 6.

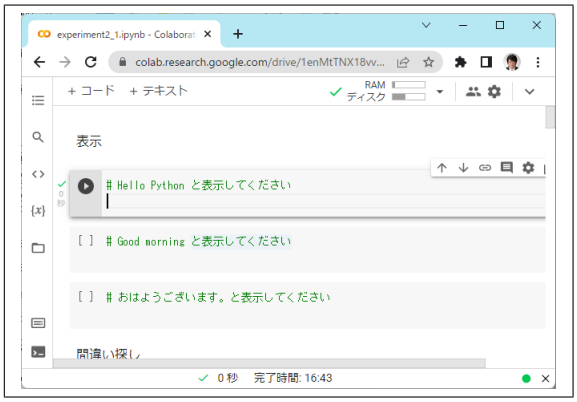

Figure 6. Python Screen on Google Colaboratory

5. Experimental Method

5.1. Outline of the Experiment

In this experiment, the participants were divided into two groups to investigate whether the intermediate content was effective in learning text-based programming. Group A experimented with intermediate content between visual and text-based languages. Additionally, Group B conducted the experiment with and without sandwiching the intermediate content for comparison with Group A. The flow of the experiment is shown in Figure 7. Nine people participated in Group A, and 12 people participated in Group B. Group A solved puzzles, mazes, and music problems. in the Blockly Games experiment. However, Group B played puzzles and mazes in the first Blockly Games experiment, and music in the second Blockly Games experiment. The details of the task are shown in the following sections.

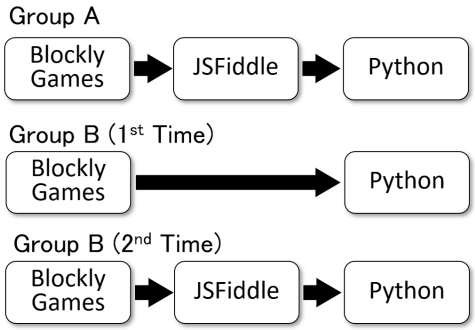

Figure 7. Experimental Flow

5.2. Experiments Using Visual-based Language (Blockly Games)

Blocky games were used to study the learning state using a visual-based language, as shown in Section 4.1. Participants were asked to answer 4 of 4 questions about puzzles, 10 of 10 questions about mazes, and 9 of 10 questions about music. Regarding music, the 10th question was omitted because of the problem of freely creating music. Here, the experimenter did not explain the operation method because the system of Blockly Games gave a lecture on how to use it. If the participants did not understand the problem or its meaning, the experimenter explained it.

5.3. Experiments Using Intermediate Content (JSFiddle)

In the experiment using the intermediate contents, JSFiddle (Section 4.2) was used. Groups A and B conducted the same experiment. Questions, including playing a C major scale, repeating music using a for statement, conditional branching using an if statement to play music, and incorporating an if statement in a for statement (2 questions), were prepared. They were asked to answer a total of six questions about reproducing the music answered in Blockly Games in the previous section of JSFiddle. Here, the experimenter gave a lecture on how to use JSFiddle and how to write a program. Next, they were asked to work on the problem. When participants did not understand, they were asked to reference the materials or ask the experimenter directly.

5.4. Experiments Using Text-based Language (Python)

In the experiment using the text-based language, Python was used, as mentioned in Section 4.3. Here, there were 30 questions and a time limit of 20 min. Participants answered questions on basic programming. Specifically, they were asked to answer questions, such as the basic usage of print statements and variables, for statements, if statements, and replacing if statements with for statements. Different questions were prepared for Group B with the same difficulty in the first and second sessions. The second Python question in Group B was the same as the Python question in Group A. Participants were asked to see the reference materials or skip the questions they did not understand. The number of correct answers was used for evaluation.

6. Evaluation of the Number of Correct Answers in a Text-based Language

6.1. Experimental Result

Participants (PART) answered 30 questions using the text-based language Python in 20 min. The number of correct answers to the question is shown in Tables 1 and 2. As the data of Participant 18 were not collected, his data were excluded from the analysis target.

From Tables 1 and 2, comparing the NCA in Group A with intermediate content and that in Group B without intermediate content shows that the NCA in Group A is higher. Additionally, even within Group B, comparing the NCA with and without intermediate content shows that the NCA in the former was higher. Especially, the introduction of the intermediate content increased the level of comprehension of the text-based language. Similar scores were found when comparing the NCA in Groups A and B with the intermediate content. In particular, this does not imply that Group B has a poor understanding. These things will be examined statistically in the next section.

Table 1. Number of Correct Answers (NCA) in Text-based Language Python in Group A

Group A	NCA
Participant 1	25
Participant 2	25
Participant 3	26
Participant 4	28
Participant 5	25
Participant 6	26
Participant 7	22
Participant 8	28
Participant 9	27
Avg.	25.78

Table 2. NCA in Text-based Language Python in Group B

Group B	1st NCA	2nd NCA
Participant 10	26	30
Participant 11	24	30
Participant 12	24	28
Participant 13	25	29
Participant 14	29	29
Participant 15	22	29
Participant 16	14	16
Participant 17	21	24
Participant 19	21	29
Participant 20	20	24
Participant 21	21	20
Avg.	22.45	26.18

6.2. Analysis of Experimental Results

A test was performed to verify the description in the previous section. First, we tested whether there was a difference in the average NCA in the text-based language Python of groups A and B. An

F-test was performed to determine if there was a difference in the variance of the NCA in Groups A and B. The results of the F-test are shown in Table 3. When the F-test was performed at a significance level of 5%, $p = 0.024$ and $p = 0.008$ were obtained in Groups A and B(first time), and Groups A and B(scond time), respectively. In both cases, $p < 0.1$ and the variances were not equal.

From this result, a t-test was performed using two samples assuming that the variances were unequal. The results of the t-test are shown in Table 4. The one-sided p-value of the significance level 5% of the average NCA in Groups A and B (first time) was $p = 0.012(< 0.05)$. Therefore, there was a difference in the average value of the NCA for groups A and B (first time). Next, the p-values on both sides of the significance level 5% of the average NCA in Groups A and B (second time) was $p = 0.796(> 0.05)$. Hence, there was also a difference in the average NCA in Groups A and B (second time).

Additionally, to confirm whether the difference in the average NCA for the first and second time in Group B was statistically significant, a two-sided t-test was performed at the significance level of 5%. As the participants were the same, a paired t-test was performed, and the p-value on one side was $p = 0.001(< 0.05)$. Therefore, it was concluded that there is a difference between the average values of the Group B (first time) and Group B (second time). A graph representing these results is shown in Figure 8.

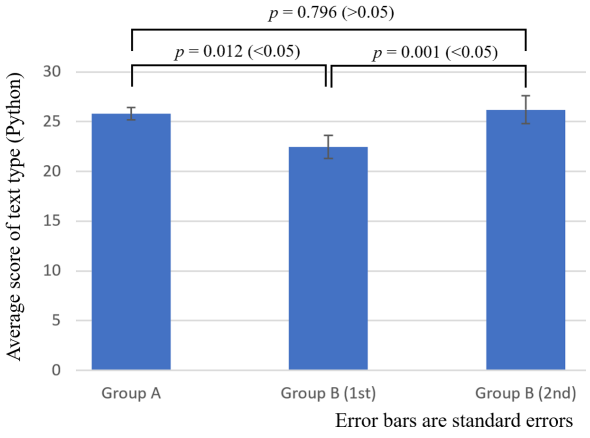

Figure 8. Average Score Of Text-Based Language (Python) For Each Group

From these results, it was confirmed that the above assumption, i.e., the understanding of the text-based language, improved by inserting the intermediate content. Moreover, the level of understanding in Group B was not lower than that in Group A.

Note that the total learning time in the experiment differs between Group A and Group B (1st). In particular, Group A has a longer total learning time due to learning with intermediate content. Concern remains that this may affect the score that expresses comprehension. A rigorous experiment in which there is no difference in the total learning time is a future issue.

7. Evaluation by Questionnaire

A questionnaire was administered after the experiment to confirm the characteristics of the intermediate content of our proposal.

7.1. Questionnaire Items

The questionnaire consisted of the following questions. All the answers had four levels of response.

Question 1: Did you enjoy creating music with JSFiddle?

Answer 1–1: I enjoyed it.

Answer 1–2: I enjoyed it a little bit.

Answer 1–3: I had little enjoyment.

Answer 1–4: I did not enjoy it.

Question 2: Do you think there are many English words (e.g., play) to memorize when using the music program in JSFiddle?

Answer 2–1: There are very few English words to memorize.

Answer 2–2: There are a few English words to memorize.

Answer 2–3: There are many English words to memorize.

Answer 2–4: There are several English words to memorize.

Question 3: Is it possible to immediately verify the results of the music program (e.g., can you check the sound immediately after execution?) in JSFiddle?

Answer 3–1: I can quickly check the execution results.

Table 3. F-test results

Group	p-value	Note
A and B (1st)	0.024(< 0.1)	Variances were not equal
A and B (2nd)	0.008(< 0.1)	Variances were not equal

Table 4. t-test results

Group	p-value	Note
A and B (1st)	0.012 (<0.05)	A significant difference
A and B (2nd)	0.796 (>0.05)	No significant difference
B (1st) and B (2nd)	0.001 (<0.05)	A significant difference

Answer 3–2: I can somewhat quickly check the execution results.

Answer 3–3: I cannot somewhat quickly check the execution results.

Answer 3–4: I cannot quickly check the execution results.

Question 4: Is it easy to hear where the notes are out of tune in the music program by JSFiddle?

Answer 4–1: It is easy to hear where the notes are out of tune.

Answer 4–2: It is a little easy to hear where the notes are out of tune.

Answer 4–3: It is not easy to hear where the notes are out of tune.

Answer 4–4: It is not at all easy to hear where the notes are out of tune.

7.2. Questionnaire Results

Upon completing the experiment, we shared questionnaires to get reviews on the characteristics of the intermediate contents. The results are shown in Table 3. The results of the questionnaire revealed that the content of the programming class for composing music with JavaScript has intermediate characteristics between the visual- and text-based programming languages.

Figure 9 shows the results of the questionnaire. A summary and interpretation of the results are shown in Table 5.

7.3. Comparison of Questionnaire Results

In our previous research (Umezawa et al, 2022), classes were conducted using the intermediate content explained so far for students of the high school attached to Shonan Institute of Technology. The same questionnaire was administered. In this section, the questionnaire results of university students andwere compared with high school students. Figure 10 shows the results of the questionnaire for high school students. Table 6 shows a comparison of questionnaire results.

Table 5. Questionnaire Results

Question No.	Mode	Interpretation
Question 1	2	They enjoyed it a little.
Question 2	2	English words to memorize are somewhere between “few” and “many.”
Question 3	1	Confirmation of execution results is between “quickly” and “somewhat quickly.”
Question 4	2	It is a little easy to hear where the notes are out of tune.

Looking at these, it can be seen that high school students had more fun than college students in terms of enjoyment (Question 1). Moreover, regarding the number of English words that must be memorized (Question 2), many students answered that high school students had to memorize many words because they did not know the number of words that text-based programming languages had to memorize. I would like to conduct further experiments targeting programming beginners of various age groups, including junior high school students. Additionally, regarding Questions 3 and 4, it can be seen that both college students and high school students show similar tendencies.

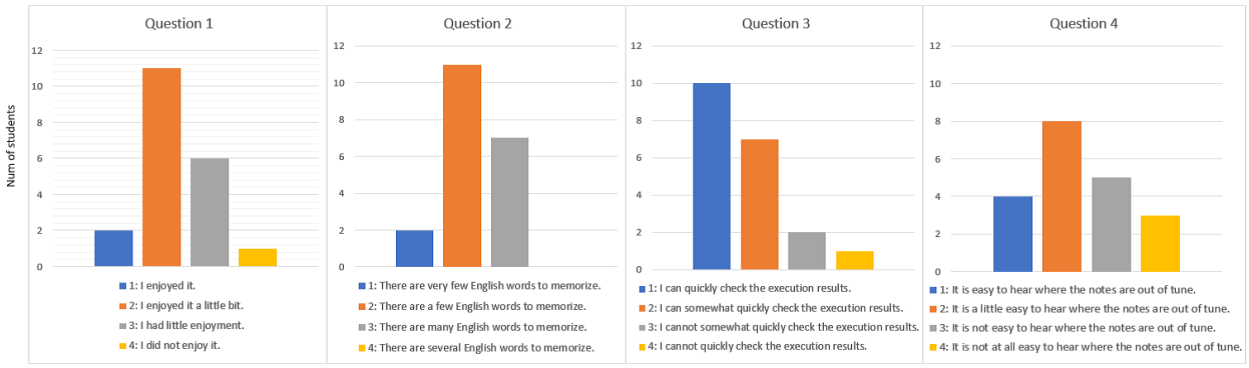

Figure 9. Questionnaire Results for College Students

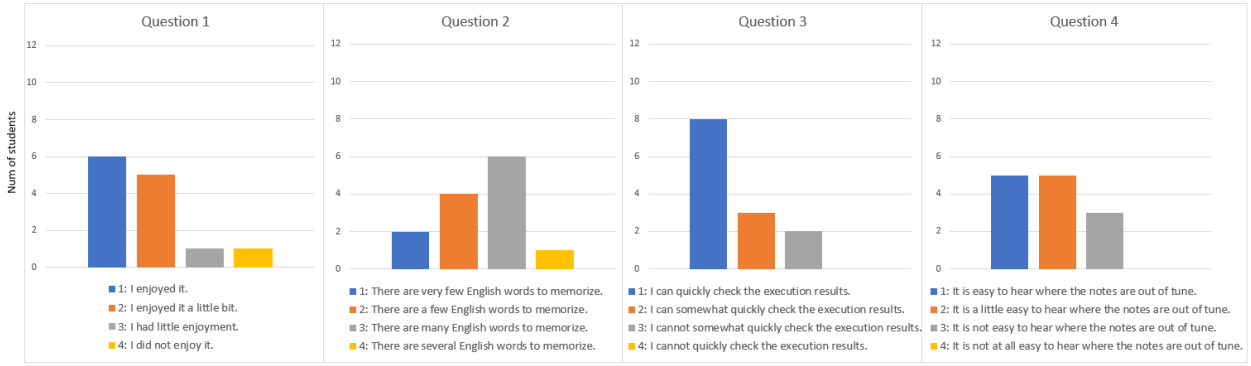

Figure 10. Questionnaire Results for High School Students

Table 6. Comparison of questionnaire results

Question No.	Mode	
	College	High School
Question 1	2	1
Question 2	2	3
Question 3	1	1
Question 4	2	1 and 2

8. Summary and Future Work

This research demonstrates how using intermediate materials between visual- and text-based languages can help learners better understand text-based languages. The results from our questionnaires showed that the proposed learning content has intermediate characteristics between the visual- and text-based languages.

As a future task, we intend to increase the number of participants in experiments and perform analysis to achieve the overall goal of our research project. We intend to examine the learning state using biological data, such as brain waves during learning, in addition to examining the results. We also intend to apply our proposal to actual classes.

About Research Ethics

The Research Ethics Committee of Shonan Institute of Technology has approved these experiments. We have also received consent to participate in this experiment from the participants and their parents.

Acknowledgments

Part of this research was conducted as part of the research project “Research on e-learning for next-generation” of the Waseda Research Institute for Science and Engineering, Waseda University. Part of this work was supported by JSPS KAKENHI Grant Numbers JP22H01055, JP21K18535, JP20K03082, and Special Account 1010000175806 of the NTT Comprehensive Agreement on Collaborative Research with the Waseda University Research Institute for Science and Engineering. Research, leading to this paper,

was partially supported by the grant funded by the ICT and Education of JASMIN research working group.

References

[1] Mason, D., & Dave, K. (2017). Block-based versus flow-based programming for naive programmers, *2017 IEEE Blocks and Beyond Workshop (B&B)* , pp. 25–28. doi: 10.1109/BLOCKS.2017.8120405

[2] Robinson, W. (2016). From scratch to patch: Easing the blockstext transition, *In Proceedings of the 11th Workshop in Primary and Secondary Computing Education (ACM)*, pp. 96–99.

[3] Mladenović, M., Boljat, I., & Žanko, Ž. (2018). Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level, *Education and Information Technologies 23(4)*, pp. 1483–1500.

[4] Navarro-Prieto, R., & Cañas, J. J. (2001). Are visual programming languages better? The role of imagery in program comprehension, *International Journal of Human-Computer Studies*, Volume 54, Issue 6, pp. 799–829.

[5] Xu, Z., Ritzhaupt, A. D., Tian, F., & Umapathy, K. (2019). Block-based versus text-based programming environments on novice student learning outcomes: A meta-analysis study, *Computer Science Education*, pp. 177–204.

[6] Daskalov, R., Pashev, G., & Gaftandzhieva, S. (2021). Hybrid Visual Programming Language Environment for Programming Training, *TEM Journal*. Volume 10, Issue 2, pp. 981–986.

[7] Weintrop, D., & Wilensky, U. (2017). Between a Block and a Typeface: Designing and Evaluating Hybrid Programming Environments, *IDC '17: Proceedings of the 2017 Conference on Interaction Design and Children*, pp. 183-192.

[8] Weintrop, D. (2015). Blocks, text, and the space between: The role of representations in novice programming environments, *In 2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 301–302.

[9] Alrubaye, H., Ludi, S., & Mkaouer, M. W. (2019). 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.

[10] Tóth, T., & Lovászová, G. (2021). Mediation of Knowledge Transfer in the Transition from Visual to Textual Programming, *Informatics in Education*. DOI 10.15388/infedu.

[11] Weintrop, D. & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms, *Computers & Education*, Volume 142, 103646.

[12] Umezawa, U., Saito, T., Ishida, T., Nakazawa, M., & Hirasawa, S. (2018). Learning state estimation method by browsing history and brain waves during programming language learning, *Proceeding of the 6th World Conference on Information Systems and Technologies (World CIST 2018)*, p.p. 1307–1316.

[13] Umezawa, K., Saito, T., Ishida, T., Nakazawa, M., & Hirasawa, S. (2020). Learning-state-estimation Method using Browsing History and Electroencephalogram in E-learning of Programming Language and Its Evaluation, *Proceeding of the International Workshop on Higher Education Learning Methodologies and Technologies Online (HELMeTO 2020)*, pp.22-25.

[14] Umezawa, K., Nakazawa, M., Kobayashi, M., Ishii, Y., Nakano, M., & Hirasawa, S. (2021). Comparison Experiment of Learning State Between Visual Programming Language and Text Programming Language, *Proceeding of the IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE2021)*, pp. 1-5.

[15] JSFiddle, https://jsfiddle.net/

[16] Blockly Games, https://blockly.games/

[17] GoogleColaboratory, https://colab.research.google.com/notebooks/

[18] Umezawa, K., Nakazawa, M., & Hirasawa, S. (2022). Seamless transition from visual-type to text-type languages, *Proceedings of the 84th National Convention of IPSJ*, Vol. 4, pp. 519–520.