2025 IEEE/SICE International Symposium on System Integration (SII)
January 21-24, 2025, Munich, Germany

Development of a Computational Thinking Learning Tool Using a Railway Toy and Music

T. Ozawa, and M. Matsumoto, *Member, IEEE*

Abstract— **In Japan, programming education in elementary schools has begun in 2020, and it is necessary to foster programming thinking effectively. Various learning tools have been developed to introduce programming concepts, and many studies have been conducted to develop tools using music to foster computational thinking that includes programming thinking. In this study, we focused on the affinity between the basic movements of the railway toy and the melodies and chord progressions, and we developed a learning tool to foster computational thinking by combining music and the railway toy. In this study, an activity using this tool was developed and subjects were asked to experience the activity. Based on the subjects' progress in the activity and their opinions in the questionnaire, the improvement of the activity is discussed.**

I. Introduction

Programming education in elementary schools has become compulsory in Japan since 2020. According to the "Guide to programming education in elementary schools"[1] published by Ministry of Education, Culture, Sports, Science and Technology (MEXT), programming education in elementary schools requires programming thinking, "logical thinking that considers how to combine given methods or how to improve them to realize activities".

Wing [2] defines the concept of computational thinking as "a thought process for formulating a problem and expressing its solution in a way that a computer (human or machine) can effectively execute". According to Computing At School (CAS) [3], an organization that supports the computing curriculum in the United Kingdom, computational thinking is located as the core of computing, and the thought processes of computational thinking include Abstraction–choosing how to represent a system to make a task easier, Decomposition–making detailed tasks dividing the original task, Algorithmic thinking–defining the order and rules of problem solving, Evaluation–checking whether the solution meets the objective, and Generalization–quickly solving new tasks based on past task solutions.

Bando et al. [4] stated that programming thinking is limited to the programming component of computational thinking, and they are concerned that emphasizing programming thinking will have a negative impact on programming education in Japan, in that it will be limited to learning the process of programming. Therefore, it is recommended to develop learning tools that foster computational thinking rather than programming thinking.

Some studies have reported the development of learning tools to foster computational thinking, especially learning tools that introduce music. Repenning et al. [5] suggested that combining music and computational thinking can simultaneously introduce activities in music education and improve computer science and music skills at the same time. However, the effectiveness of improving programming skills has not been demonstrated. This could be because the developed tool requires block-based programming on a PC using Scratch, which is a high level of difficulty for those who have not yet experienced programming on a PC. However, Sim et al. 2021[6] found that students who have no experience with programming languages can foster computational thinking when using an unplugged tool.

Several studies have been conducted on programming activities using toy trains. In particular, in Japan, there are studies on programming activities using Plarail, a toy train manufactured by TOMY Corporation. Kawanami et al. [7] reported that the reason for using Plarail is that it is a popular toy among many children, including girls, and that the results are easy to recognize visually, making it easy to perform trial-and-error programming. Thus, although there are some studies on the use of railway toys, we could not confirm any studies that combined music and railway toys. In addition, as in the study by Sasamoto et al. [8], it is possible to realize an unplugged tool using railway toys.

Currently, we could not find any study regarding the affinity of music and railway toys. However, in this study, we expect the affinity between railway toys and music from several correspondences. For example, the rails in railway toys correspond to the layout of musical scores in music. The running railway toys is associated with the performance part of the music. We believe that the ease of correspondence between railway toys and musical scores indicates the affinity between railway toys and music.

Therefore, we focused on the affinity between running trains and melodies or chord progressions, and we developed a music learning tool [9] to foster computational thinking, in which the tracks of a railway toy are used as musical notation by mapping a train running to musical elements.

In this paper, we present detailed systems of the developed unplugged tool and an example of its activities. An unplugged tool means a tool that does not involve any coding on the PC. It also discusses programming concepts and computational thinking that can be experienced when using the unplugged tool, and additional programming concepts fostered by the

T. Ozawa was with the University of Electro-Communications, 1-5-1, Chofugaoka, Chofu-shi, Tokyo, 182-8585, Japan

M. Matsumoto is with the University of Electro-Communications, 1-5-1, Chofugaoka, Chofu-shi, Tokyo, 182-8585, Japan (corresponding author e-mail: mitsuhaur.matsumoto@ieee.org).

specific playing method. We think that not only users who do not have computational thinking can acquire the ability, but also users who have some degree of computational thinking can improve their abilities by repeatedly using the proposed system. We developed the proposed system the aim of creating a tool that can enhance the thinking skills of both the former and the latter.

II. Related Works

A. Computational Thinking Activity with Music

Baratè et al. [10] proposed Lego Music Notation, which represents musical notation using Lego bricks as musical notes and symbols. They introduced a simplified representation of music by mapping percussion notation, chord representation, and multi-member playing with Lego bricks, and introduced two types of activities as examples of Lego Music Notation tasks: an activity to translate from Lego bricks to notation, and an activity to translate from notation to Lego bricks.

Repenning et al. [5] tested the effectiveness of a learning course for computational musical thinking, that combines music and computational thinking. They defined five operational concepts of computational musical thinking - Interpreter, Chance, Interaction, Hierarchy, and Rewrite-rules and proposed a PC-based music creation tool based on these concepts.

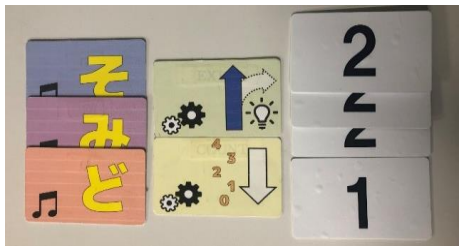
Note Code by Kumar et al. [11] is a puzzle-type music tool for developing computational thinking. The tool controls the output music by operating and combining box-shaped blocks. The programming concepts used in this tool were the branch statements and function modules. The tool is an unplugged tool, but it has created a feature that enables the output of the tool to be displayed on a PC simultaneously.

B. Programming Activity with Railway Toy

Tosten et al. [12] created an activity that controls the movement of trains on tracks to avoid train crashing. This activity creates control commands by combining predicate-based programming of the track switch, train movement, and coupling. The player can also debug an activity through a crash.

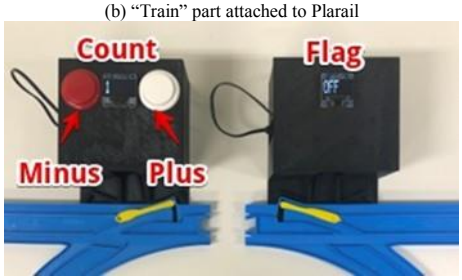
Kawanami et al. [7] created programming educational materials for junior and senior high school students to learn the basic concepts of programming and electronic engineering using the PC software Arduino to control Plarail. The results of the experiments on subjects reported that Plarail is easy to interest and that they can easily maintain their attention because they use real objects.

Sasamoto et al. [8] proposed an unplugged railway programming toy that simulates computer concepts, such as copying, adding, subtracting, and comparing. The proposed toy is a combination of Plarail and balls and can control the train according to the balls loaded on the train, using the Plarail tracks as the program code and the balls as variables. When a train behaves unexpectedly, it can change its behavior by stopping the train and changing the loaded balls.

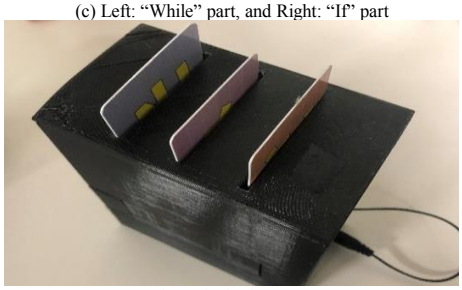


(a) Left: Music card, Center top: Exist card, Center bottom: Count card, and Right: Variable card

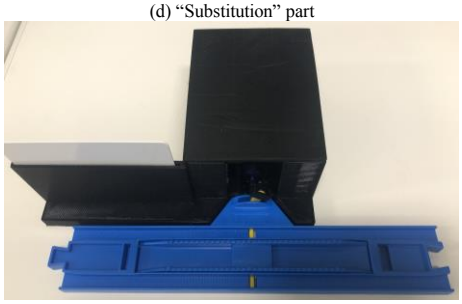
(b) "Train" part attached to Plarail



(c) Left: "While" part, and Right: "If" part



(d) "Substitution" part



(e) "Module" part

Figure 1. Expansion parts of the proposed tool

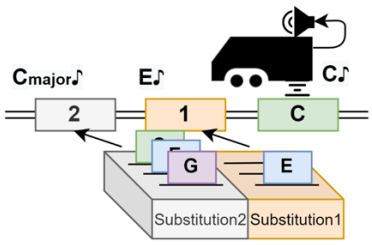


Figure 2. The system of generating sounds

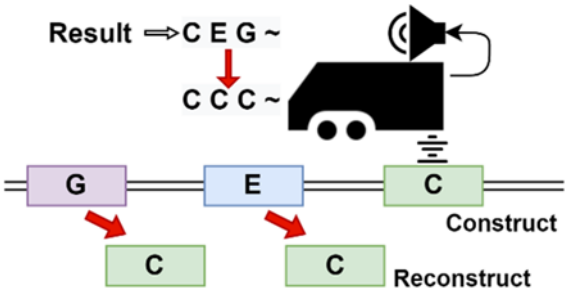


Figure 3. The system of outputting music and reconstructing cards

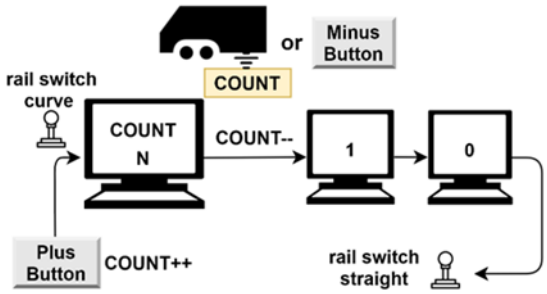


Figure 4. The system of repeated statement

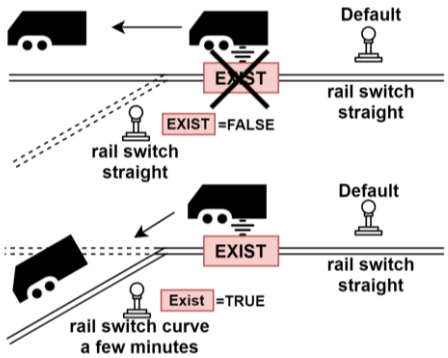


Figure 5. The system of branch statement system

III. Details of the Tool and the Activity

A. Tool Concepts

The developed tool introduces an intuitive system that uses a railway toy to produce music as the output. We created expansion parts to realize programming concepts and music output. We have also created a learning tool that is an extension of the railway toy, with the aim of learning musical scores that are difficult for inexperienced users to learn. We also designed some activities using the tool.

We also created the fostering computational thinking activity by constructing railway toys and expansion parts for the main purpose of abstracting music notation, which is too complex for inexperienced users.

The target age group of the proposed tool is 6 years and above, with no upper limit. The minimum age is 6 years old because it is intended to be used as an introduction for pre-school to primary school children. There is no upper limit because it is intended to be played with children and to be used as an introduction to programming, even for older children with no programming experience.

We think of music and Plarail movement as complementary feedback rather than independent feedback. When giving feedback on whether the programming as a whole is working or not, we consider the two elements - the flow of execution (Plarail) and what is happening in that part (music) - to be two important elements.

The developed tool's system includes three main points. Figure 1 shows examples of expansion parts introduced in the following sections.

B. Player Operations and Sound Output System

The proposed system consists of several expansion parts and cards that can be attached to and removed from Plarail. As shown in Fig. 2, the player designs the desired music based on the layout of Plarail tracks and cards placed on the tracks. After the design, when the train is running, sound is generated each time the train reads the card placed on the tracks. The leading locomotive is attached to the "train" part, which is a reading unit of cards. When a music card or variable card corresponding to the substitution part is read, a sound corresponding to the card is generated. music card is used to generate the sound that is painted on it, and variable card is used to generate the sound of the music card inserted in the specified substitution part, such as a program variable. Substitution part also enables the operation to change the sound on the track to a different sound all at the same time. As shown in Fig. 3 when the music card or variable card is reconstructed, the train reads a different card and outputs different music than before. We expect that variable cards are particularly useful for expressing chords. If users want to place a 3-note chord in N places, they will need $3N$ cards if they do not use variable cards. On the other hand, if users use variable cards, they can set 3 sounds as variables and set variable cards in N places, so users only need to install N+3 cards. This process is similar to the role of variables in program procedures, and we expect that it will help users understand the concept of variables.

C. Loop Statement System

"While" part and count card are used to realize the loop statement. The direction of the branch track is switched based on the remaining count in "while" part. As shown in Fig. 4, when the remaining count is one or more, the branch track is switched to the curved direction, and when the remaining count is zero, the branch track is switched to the straight direction. In addition, when the "train" part reads a count card on the track, the remaining count on "while" part decreases by one. Pushing the plus button in "while" part once increases the remaining count by one and pushing the minus button once decreases the remaining count by one.

D. Conditional Statement System

"If" part and exist card are used to realize the conditional statement. The direction of the branch track is switched by the flag of "if" part. As shown in Fig. 5, when the flag is True, the

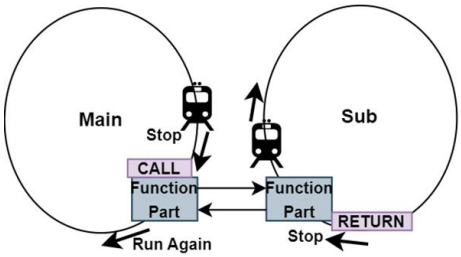


Figure 6. The system of module function

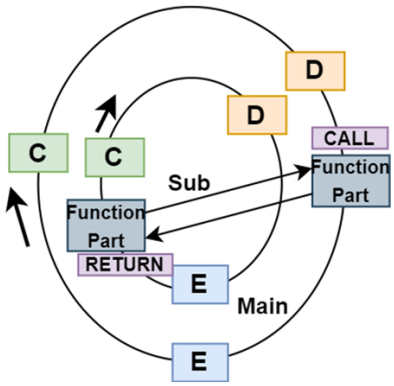


Figure 7. The system of recursion processing

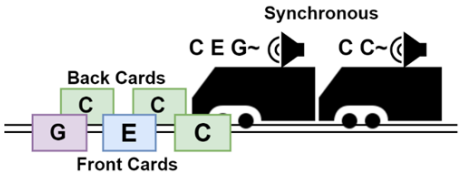


Figure 8. The system of parallel processing

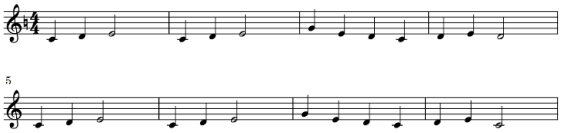


Figure 9. Notation of Japanese children's song Tulip.

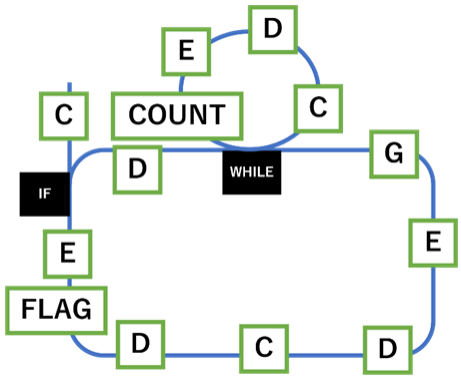


Figure 10. Layout 1 to generate Tulip

branch track switches to the curved direction, and when the flag is False, the branch track switches to the straight direction. When the "train" part reads an exist card on the track, the

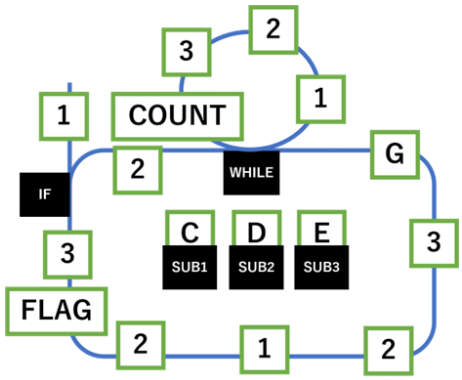


Figure 11. Layout 2 to generate Tulip

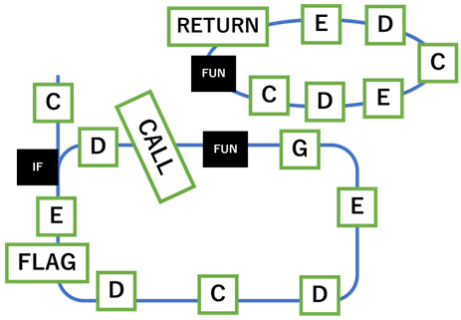


Figure 12. Layout 3 to generate Tulip

branch track is switched to the curved direction. After a few seconds, the branch track returns to the straight direction.

E. Module Function System

The representation of module function is seen as a function call in the process of the main program function. As shown in Fig. 6, two tracks are created: one track is the main program function and the other is a sub program function, and module part is placed on each track. "Module" part is used to control the running and stopping of the train in relation to the other "module" part. When the train stops at the "module" part of one track, the train at the "module" part of the other track runs. When the train stops at the "module" part on the main-track, the train on the sub-track runs, and when the train stops at "funmodule" part again after a round of running on the sub-track, the train on the main track runs again. Thus, it is possible to reproduce program execution in which a sub-function is called by the main function, the sub-function processes and returns, and the main function processes again.

F. Recursion System

"Function" parts are used to represent recursion. To introduce recursion into a melodic structure, first prepare a phrase and then insert the phrase between any of the phrases. For example, if you want to express recursion using the phrase CDE, insert CDE between CD and E. In this case, the output melody is CD CDE E. On the basis of this system, the recursion is expressed using "function" parts. Prepare two closed rails with "function" parts installed, as shown in Fig. 7, and place another closed rail inside one closed rail in a doughnut shape. Place a calling card immediately before the "function" part on the outer closed rail and a return card immediately before the

STATUS OF IMPLEMENTATION OF RAIL PART CONSTRUCTION FOR ACTIVITY TASK 2 AND 3

Number of Experiment	Task 1. Sound Output	Task 1. Loop	Task 1. Conditional	Task 1. Substitution	Task 1. Call Function	Task 1. Recursion	Task 1. Pallarel Processing	Task 2. Music Creation with Tools	Task 3. Players Create Their Own Music
1	○	○	○	○	○	○		○	
2	○	○	○	○				○	
3	○	○	○	○				○	○
4	○	○	○	○	○			○	
5	○	○	○	○	○			○	
6	○	○		○	○		○		
7	○	○	○	○				○	
8	○	○	○	○				○	○

inner “function” part. Next, place the inner CDE note card from the “function” part all the way round to where the return card is to be read, and on the outside place the CD card call card up to where the return card is to be read, and place the E card after the “function” part. In this way, the outer vehicle part outputs CD, then CDE on the inside, then E again on the outside, thus outputting a recursive melody structure (Recursion). In addition, by installing closed rails with the same note cards on the inner and outer sides, as in a doughnut shape, it is expected that the player will be able to recognize the form of two closed rails with the same structure calling and being called by each other, making it easier to visually recognize the recursion.

G. Parallel Processing System

The representation of parallel processing is seen as an ensemble performance that simultaneously plays multiple parts as shown in Fig. 8. To achieve this, two “train” parts are attached to the Plarail. The cards read by the first “train” part is placed on either side in the direction of running, and the cards read by the second “train” part is placed on the opposite side. Thus, when the train runs, the music corresponding to the cards placed on both sides is simultaneously outputted. Parallel processing can be implemented by preparing two trains and “train” parts connected to each, and running each. The train speed can be changed with the Plarail power switch.

IV. ACTIVITY THAT CREATES MUSIC USING THE TOOL

A. Activity Overview

We developed some activities that uses learning tools to foster computational thinking and programming concepts.

The following is an overview of the entire activity. The activity consists of three components: Task 1: "Video on how to use the learning tool," Task 2: "Create music using the tool," and Task 3: "Players create their own music". The activity can be performed from any of the three tasks, and it is not necessary to perform all three. The difficulty level increases in the order of Task 1, Task 2, and Task 3. By adjusting which task to perform and which task to give priority to, the player can adjust the difficulty level according to their own level of proficiency.

The activity uses Plarail, parts, and a tablet. The animation of Task 1 and the example layout of Task 2, which will be explained later, can be viewed on the tablet.

B. Task 1: Watching a Video on How to Use Learning Tools

We have the player watch a video that explains how to play the tool and how it works in this task. Seven videos (including 8 programming concepts) were created: "Sound output", "Sound substitution" (including substitution and variable), "Branching", "Repeat", "Function call", "Recursion", "Parallel processing”. The videos are subtitled with explanatory text that describes the process of constructing the rails and parts, as well as the actual sound output and changes in the rails and parts after construction. Players can actually construct the tool while watching the video and checking the flow of the play. Players can adjust their level of activity according to their interest and skill level, such as actually constructing the tool while watching the video or watching the video only.

C. Task 2: Music Creation with Tools

The player builds rails and parts in order to output the music for the task. To assist players in constructing the music for the task, several examples of rail/part layout for creating the music for the task, and players who need them can build them with checking the layouts. The melody structure of Tulip is shown in Figure 9.

As shown in Figure 10, the first layout uses while and “if” parts. In the CDE CDE part of the tulip lyrics, the same melody phrase is repeated twice. This part can be represented by a loop. Specifically, the phrase CDE can be repeated twice by increasing the remaining count to 2 using the red plus button attached to the “while” part, and inserting CED music cards and a count card in the closed rail following the curve direction of the loop part. After the first CDE CDE, the remaining count is zero, so the player pushes the plus button twice to reset the remaining count to 2 while the train is traveling on the part corresponding to GEDC DED.

The GEDC DED/DEC part in the tulip lyrics has a different melody phrase at the last D and C, and this part can be represented by a branch. In the first round of branching, the rail remains the curve direction by not placing a flag card just before “if” part. On the other hand, at the second branch, while the train is running, a flag card is placed just before “if” part, causing the rail to switch to the straight direction. By changing the destination of the rail between the first and second branches, a partially different melody can be expressed.

The second layout shown in Figure 11 uses substitution parts in addition to the while and “if” parts of the first layout. The C card is inserted in substitution part 1, D is inserted in substitution part 2, and E is inserted in substitution part 3. Also,

prepare variable cards 1, 2, and 3, and replace the C card on Layout 1 with variable card 1, D with variable card 2, and E with variable card 3.

The third layout shown in Figure 12 differs from the first and second layouts in that it uses if and "function" parts. The CDE CDE part is considered a sub-function and the GEDC DED/DEC part is considered the main function to introduce function calls. Player prepares one closed rail (sub rail) and another closed rail (main rail) containing "if" part. On the sub rail, place a "function" part and a return card immediately before the "function" part, and place the CDE CDE note cards in sequence. Similarly, on the main rail, place a note card, a "function" part, and a call card immediately before the "function" part.

D. Task 3: Players Create Their Own Music

In addition to the music in Task 2, players are free to install rails and parts in order to output any music they wish to implement. One major difference from Task 2 that the player attempts to output music while checking layout hints to create the music for the task, as necessary is that in Task 3, the player must imagine the melody they want to output without layout hints, and build a structure that fits the melody from scratch, while in Task 2. In Task 3, you need to imagine the melody you want to output without layout hints and build a structure that fits the melody from scratch.

V. SUBJECT EXPERIMENTS

We conducted an experiment to verify whether the activities could foster computational thinking and help students learn programming concepts. Although the tool is intended for older children (6 years old and up), we tested whether it is also effective for college students. In addition, we conducted a questionnaire in this experiment to collect opinions for improving the tool.

A. Method of the Subject Experiment

In the subject experiment, a questionnaire was administered to collect positive and negative opinions of the activity after its implementation.

A total of eight subjects participated in the experiment. Regarding their programming habits, six subjects did little programming during the week and two subjects did programming continuously. Three subjects stated that they were doing informatics research, and five subjects stated that they were non-informatics or could not decide whether they were informatics or not.

In the activities, subjects undertook activities of section 4. Subjects were asked to perform the activities for approximately 45 minutes (±5 minutes) and to finish the activities when the time limit approached. Information was collected on what kind of layout they created when they created the layout of the rail section of the task piece in task 2 and freely created the layout in task 3.

B. Subjects' Performances in the Activity

This section shows which tasks the eight subjects performed during the 45-minute activity. In Task 1, the eight subjects

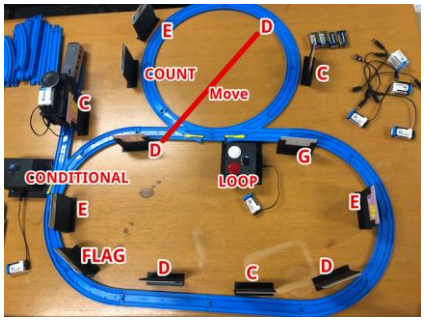


Figure 13. Layout 1 of creating "Tulip" by subject 7

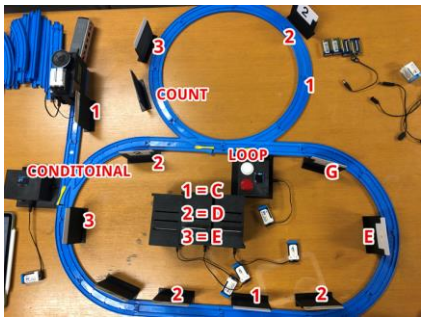


Figure 14. Layout 2 of creating "Tulip" by subject 7

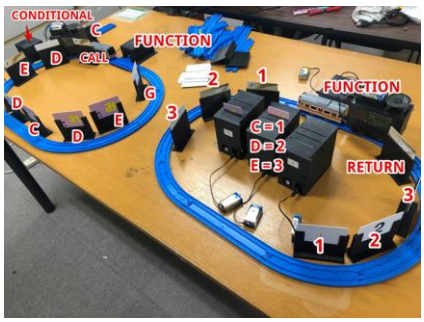


Figure 15. Layout 3 of creating "Tulip" by subject 5

watched all the videos. Next, the Table 1 shows which tasks the subjects constructed rail parts for, from Task 1 to Task 3. As shown in Table1, seven subjects performed one of the plastic rail constructions presented in Task 2, and two subjects performed up to Task 3. In particular, subject number 3 and players 7 and 8 performed multiple Task 2 or 3 constructions. Subjects 3 and 8 performed one layout construction of Task 2 and one construction of Task 3, for a total of two, while Subject 7 performed two layout constructions of Task 2.

C. Layouts Constructed by Subjects

Some of the layouts of the rail parts for creating the music actually constructed by the subjects in Tasks 2 and 3 is as follows.

Layout 1 and Layout 2 created by Subject 7 are shown in Figure 13 and Figure 14. Subject 7 created Layout 1 once and then modified it to create Layout 2 by adding substitution parts and variable cards. In the Figure 14, the substitution parts 1, 2 and 3 are assigned to C, D, and E. Note that 1 variable card is missing in the direction of the curve of the "while" part in the diagram. Subject 5 had created a layout based on Layout 3

with additional substitution parts as shown in the Figure 15. To create the layout containing the "function" parts, subject 5 realized on his own that he had not prepared enough C D E card by us, so he had to increase the number of cards by assigning a C D E to the substitution parts 1, 2 and 3, respectively, and built the layout.

VI. Discussion and Conclusion

This paper presents the details of an unplugged tool that includes programming concepts and outputs music by building railway toys and expansion parts, and introduces activities using the tool. Evaluation experiments with test subjects were carried out.

Consideration needs to be given to players who do not know how to operate the Plarail or who do not understand music (mainly pitch and notation). Although Subject 8 answered that it was the first time for him to play with Plarail, he performed the activity smoothly after the basic operations of assembling the rails and placing the train on the rails and starting the train were taught at the initial stage of the activity. From this result, it seems necessary to prepare a separate video on how to operate Plarail for players who are experiencing Plarail for the first time. In addition, some subjects did not know the melody of the tulip in the first place in the process of performing Task 2, so in addition to the Task 2, it is necessary to help them such as a video showing the actual creation of the layout of Task 2 and music sheets with the pitches rubbed on them.

Very few players actually assembled recursion and parallel processing in this study. Recursion, in particular, is a complex operation, so it is advantageous to make it interesting and guide the players to unconsciously assemble it, which will lead to an improvement in comprehension. A possible improvement in this respect is to create videos that mix two or more programming concepts in Task 1. For example, a combination of parallelism and loops, or a combination of recursion and branching. In particular, combining recursion, conditional statement and loop statement could lead to the introduction of algorithms such as search algorithms and sorting, and is expected to generate interest even among older players.

In the questionnaire, as an evaluation of the activity, all subjects gave positive answers, such as agreeing or rather agreeing, when asked whether the activity was meaningful for learning programming concepts for children aged between 6 and 9 years old from the viewpoint of the subjects. When asked if the subjects themselves found it meaningful to learn programming concepts, seven subjects responded positively, either agreeing or somewhat agreeing, while one subject responded neither agree nor disagree. The subject who answered neither described his opinion as "it was a programming concept that I already knew". Thus, some older players and players with programming experience may seek programming concepts that they do not know, requiring a further range of difficulty levels for the activity. The corresponding solution is a measure similar to the one described above, which is to provide videos of existing algorithms being implemented with this tool.

In addition, one of the comments on the activity pointed out that they would like to have information on how to improve the activity in case of errors. It is considered important to provide the player with a solution when an error is made, for the activity to run smoothly. If an error occurs when a child is playing alone and it becomes impossible for the child to continue alone, the child is likely to abandon playing. Therefore, it is necessary to add measures for each error situation in the explanatory video for each tool in Task 1.

In the future, while improving the tool, we would like to test whether the activity increases scores in computational and programming thinking skills. Implementation of parameter expressions in functions is also an issue for the future.

References

[1] MEXT: "Elementary School Programming Education Guide (3rd Edition)", https://www.mext.go.jp/content/20200218-mxt_jogai02-100003171_002.pdf (Access: Sep. 24, 2022) (in Japanese)

[2] J.M. Wing: "COMPUTATIONAL THINKING BENEFITS SOCIETY", http://socialissues.cs.toronto.edu/index.html%3Fp=279.html (Access: Sep. 24, 2022)

[3] Computing at School, "CAS computational thinking - A Guide for teachers", https://www.computingatschool.org.uk/teaching-resources/2014/june/cas-computational-thinking-a-guide-for-teachers (Access: Sep. 24, 2022)

[4] T. Bando, M. Kuroda, M. Fukui, and J. Moriyama: "A Critical Review on Institutionalization of Programming Education at Elementary and Secondary School Level in Japan", Hyogo University of Teacher Education, Center for School Education Research, No.30, pp.173-184 (2017) (in Japanese)

[5] A. Repenning, J. Zurmühle, A. Lamprou, and D. Hug: "Computational Music Thinking Patterns: Connecting Music Education with Computer Science Education through the Design of Interactive Notation", CSEDU, vol1, pp.641-652 (2020)

[6] T. Y. Sim, M. Teow, and S. L. Lau: "Unplugged Computational Thinking Activities Framework Development for Novice Programmer", 2021 IEEE International Conference on Computing (ICOCO), pp.297-302 (2021)

[7] T. Kawanami, S. Kaji, D. Takago, and R. Hayashi: "Development and Evaluation of Introductory Educational Materials for Embedded Systems using Toy Train", KIT Progress, No.24, pp.21-30 (2016) (in Japanese)

[8] H. Sasamoto, H. Noma, K. Susami, Y. Itoh, Y. Kitamura, F. Kishino, and N. Tetsutani: "Hands-on Learning of Computer Programming for Children Using a Model Railway", JSSE Research Report, Vol.18, No.4, pp.1-6 (2003) (in Japanese)

[9] T. Ozawa, M. Matsumoto: "Development of a programming learning tool using a train toy", IEEJ Transactions on Electrical and Electronic Engineering, pp.1351-1353, Vol.16, No.10 (2021)

[10] A. Baratè, L.A. Ludovico, and D.Malchiodi: "Fostering Computational Thinking in Primary School through a LEGO ® -based Music Notation", Procedia Computer Science, No.112, p-1334–1344 (2017)

[11] V. Kumar, T. Dargan, and U. Dwivedi: "Note Code: A Tangible Music Programming Puzzle Tool," Tangible, Embedded, and Embodied Interaction (2015)

[12] R. S. Tosten: "Using a model railroad system in an artificial intelligence and operating systems course", ACM SIGCSE Bulletin, Vol.25, Issue.1, pp 30–32 (1993)