Chang, M. et al. (Eds.) (2019). Proceedings of the 27th International Conference on Computers in Education. Taiwan: Asia-Pacific Society for Computers in Education

A Programming Learning System Introducing Small Steps Involving Mutual Evaluation

Hideyuki TAKADA*, Ayaka IWASA, Risa MATSUBARA, Yuki TAKEDA, Tsuyoshi DONEN
Faculty of Information Science and Engineering, Ritsumeikan University, Japan
*htakada@cs.ritsumei.ac.jp

Abstract: In this paper, we propose a programming learning system which incorporates a concept of small steps to nurture the computational thinking. While the project-based programming environment allows children to reach an unlimited goal of their achievements, most of learners often suffer from finding a meaningful project to work and coming up with a way of building programs for their project. In order to scaffold their programming experience, our system gives a step-by-step procedure to build a complete meaningful project, where the entire project is decomposed into small steps in advance. In addition, the system requires learners to ask their peer to check if an ongoing step is correctly completed before proceeding to the next step. In this way, our system allows them to experience the repeated cycle of the computational thinking process while encouraging them to interact each other. We also show our empirical findings obtained by applying this system to a programming workshop. In this workshop, 11 small steps to build a game project were provided to participants with our iPad application. As a result, we observed that they could complete the given project regardless of their programming experience and the system gave them an opportunity to interact with others while they were doing programming.

Keywords: Learning with programming, Computational thinking, Small steps, Mutual evaluation

1. Introduction

Various efforts have been made for several decades to introduce programming to children's learning activities (Papert, 1980)(Resnick & Robinson, 2017). The purpose of doing programming in primary education is not to learn programming itself to be a computer programmer, but to learn *with* programming. Through learning with programming, children are expected to acquire "21st century type skills" such as critical thinking ability and problem solving skills. These abilities have a common factor with the thinking process called "computational thinking" (Wing, 2006). Curriculums incorporating computational thinking are already applied to education in several countries including Australia (Falkner, Vivian & Falkner, 2014), the United Kingdom (Department for Education, 2013), and the United States (Grover & Pea, 2013).

Project-based programming environments such as Scratch (Resnick, et al., 2009) would be a good candidate to be used in a learning activity to nurture the computational thinking. In our experience in organizing a programming workshop for over the last 10 years, however, we often see many children suffering from finding a meaningful project to work, and having difficulty in building a program for what they want to create. In order to overcome this situation, we believe that it is very important to scaffold children's programming experience by guiding them to complete a meaningful project during a programming workshop.

Based on these thoughts, we propose a programming learning support system which enables children to develop computational thinking skills by working on a programming project which is broken down into small steps. This system gives a learning material with a step-by-step procedure to build a project on Scratch, leading children to repeating the cycle of the computational thinking. In addition, the system requires them to ask their peer to check if each of the steps is completed successfully, encouraging them to interact with others. We also present our empirical findings as a result of applying this system to a workshop for elementary school students.

The rest of this article is organized as follows. Section 2 explains the concept of computational thinking, how the mutual evaluation and breaking down of a project into small steps can be incorporated in the process of computational thinking. Section 3 introduces the proposed system and its implementation. Section 4 shows the result of a practical use of the proposed system. Section 5 summarizes the research and the remaining issues for the future research.

2. Programming Learning to Foster Computational Thinking

2.1 Computational Thinking

Computational thinking refers to a thinking process in solving problems. Figure 1 shows the process of computational thinking. Computational thinking aims to solve the entire problem by repeating a cycle of the following three stages: "formulation of a problem," "expression of a solution method," and "execution and evaluation of a solution method." It is known that an original large problem which is difficult to solve as it is can be made easily solvable by breaking it down into small problems.

We believe that programming enables children to repeatedly follow the cycle of Figure 1, leading to the development of computational thinking. In order to successfully turn the cycle of computational thinking, the problem formulation plays an important role. In order to solve a problem, it is necessary to know which part of the problem should be solved first and which part to follow. Even a large, complicated problem can be understood by disclosing it little by little.

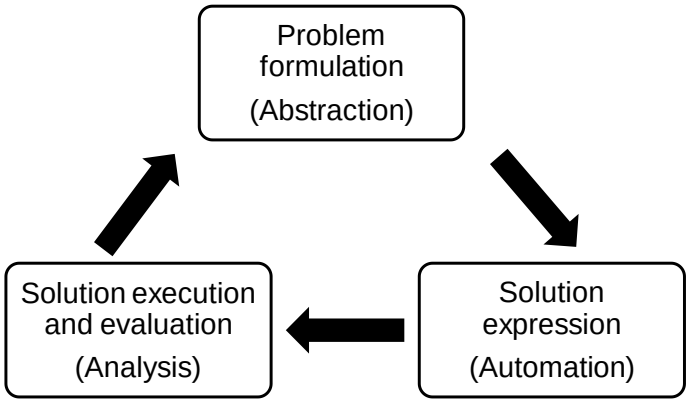


Figure 1. Computational Thinking.

2.2 Mutual Evaluation

It is also important to incorporate a chance of communication with others to a learning activity. If a mechanism to encourage students to communicate with others is introduced in the process of computational thinking, their learning experience will be a rich one.

During programming learning, checking whether programs work properly or not is very important. From the perspective of encouraging students to communicate among them, introducing a mutual evaluation into the "solution execution and evaluation" process of the computational thinking is considered to be effective.

2.3 Breaking Down a Project into Small Steps

"Small steps" are a series of tasks finely defined to resolve a meaningful problem. Breaking down a problem into small steps makes it easier to understand which part of the problem should be solved first and which part to follow. By doing so, the cycle of the computational thinking can be processed successfully. Breaking down into small steps will also provide opportunities of mutual evaluation, which also leads to an interactive learning.

Steps are not necessarily made to be more difficult or complicated as proceeding to later steps, but it is important that each step should be defined as a functional unit to complete a programming project.

2.4 Related Works

The term, *worked example*, demonstrates a step-by-step instruction of how to perform a task or solve a problem (Clark, Nguyen, & Sweller, 2011). Learning from worked examples was found to be an effective instructional strategy in such fields as mathematics, physics, and computer programming (Atkinson, et al., 2000).

We believe that one of the important factors to successfully work on a programming project is the ability of identifying functional components to achieve the target outcome. We see, however, that children often have a difficulty in dividing a whole function into several components, leading to an untidy way of doing programming. Giving them small steps to build a target project would help them understand how to identify the functional components in a real project.

We also introduce the mutual evaluation to check if each step has been completed successfully. Combined with the cycle of solving problems in the computational thinking, the mutual evaluation would play an important role in satisfying a requirement of interactive learning in programming workshops.

3. The Proposed System

3.1 Overview of the System

3.1.1 Structure of Steps

In this research, we assume that children work on a project of making a game on a graphical programming environment, Scratch. Programming on Scratch is performed by assembling blocks each of which corresponds to a specific instruction or function.

We choose a game project and break it down into small steps so that the function such as moving characters and making a hit judgement can be structured in order. An example of a step is given in Figure 2. One step consists of a functional statement (e.g. “Let the bear move to right”) to be achieved and necessary blocks (e.g. “when the flag clicked,” “forever” and “change x by 10”) to assemble.

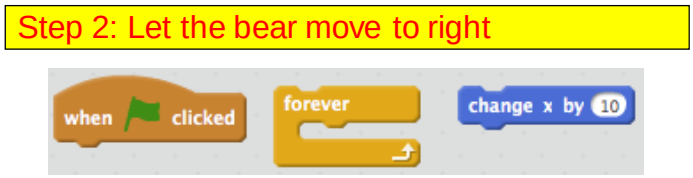


Figure 2. An Example of a Description for a Step.

3.1.2 Introduction of Mutual Evaluation

In order to enhance the programming experience working on a project with small steps, we introduce a function which allows children to proceed to the next step only after confirming the current step is completed successfully. This function aims to execute and evaluate the solution method properly and conduct an interactive learning. This function gives a screen for children to request others to check if the project is successfully completed or not. Unless other children input a password on the screen, he/she cannot proceed to the next step.

3.2 Overview of the System

This system consists of the following six screens.

- Log-in screen
- Project screen
- Test prompt screen
- Check request screen
- Clear notification screen
- Check password input screen

Figure 3 shows examples of each screen. Transitions between the screens are shown in Figure 4.

First, a user logs-in the system. Registered users can log-in from the log-in screen. Unregistered users need to register from “sign-up.”

Users can view each of the small steps separately on the screen. On the project screen, description of a step to be completed can be viewed. By pressing the “Done” button, the user proceeds to the test prompt screen. The test prompt screen gives an instruction to evaluate bt themselves if the function is correctly performed or not.

Once the self-evaluation is completed on the test prompt screen, the user can proceed to the check request screen. On the check request screen, functions to be checked and a password will be displayed. By having the password input on the check password input screen of another user’s tablet terminal, the user can proceed to the clear notification screen. When asked to check by another user, the user can proceed to the check password input screen from the project screen, the test prompt screen or the clear notification screen. The check password is randomly generated every time when the check request screen opens.

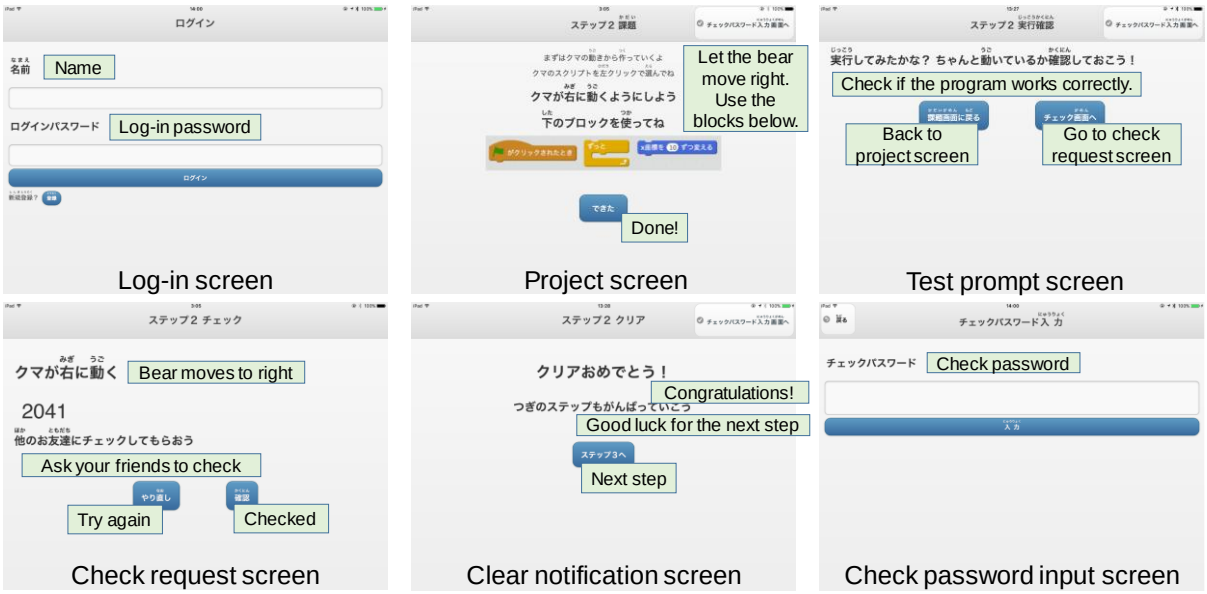


Figure 3. Screen Examples of the System.

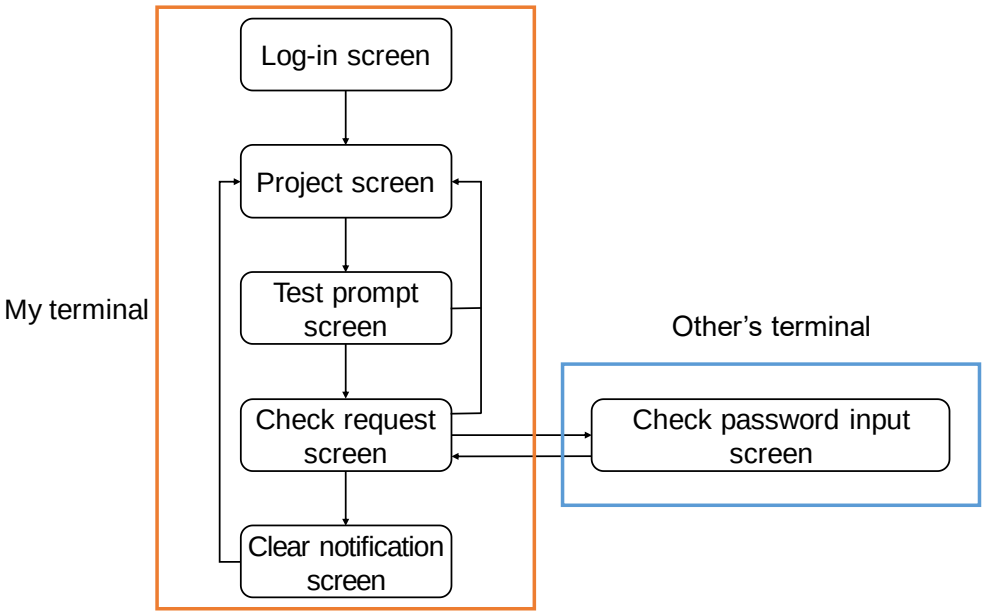


Figure 4. Screen Transitions.

3.3 Practical Implementation of the Proposed System

The system consists of a client application for browsing and checking the task steps, and a server-side application for managing the project contents and user information.

Because schools and workshops have different terminals with different operating systems, we aimed for multi-platform use and developed the system with Monaca (`https://monaca.io/`) which accommodates the multi-platform development. HTML5 and JavaScript were adopted for an application development on Monaca.

The server side environment is implemented on the NIFCLOUD mobile backend which can be integrated with Monaca. The server manages the user information and the status of checking the progress of a project using the database function.

4. Evaluation

This section explains a practical application of the proposed system to a programming workshop and shows our experiences.

4.1 The Process of Making the Proposed System into Practice

We have implemented the proposed system at a programming workshop organized by a non-profit organization called “Super Science Kids.”

The system was evaluated from its usage log, recorded videos and questionnaires conducted after the workshop. Two terminals, a laptop for programming with Scratch and Apple’s iPad for using the proposed system, are assigned to each of the participants.

4.2 Contents of the Workshop

In this workshop, participants worked on a project under a topic of “Let’s make a snow game!” The screen of this project is shown in Figure 5. This game uses a background image and sprites included in the standard installation of Scratch. In this game, a player moves the bear with the left and right arrow keys while catching snowflakes falling randomly. Every time when the bear touches the snowflake, it gets bigger. When the bear becomes big enough to touch the star, The game is completed. The bear also gets smaller repeatedly in every ten seconds.

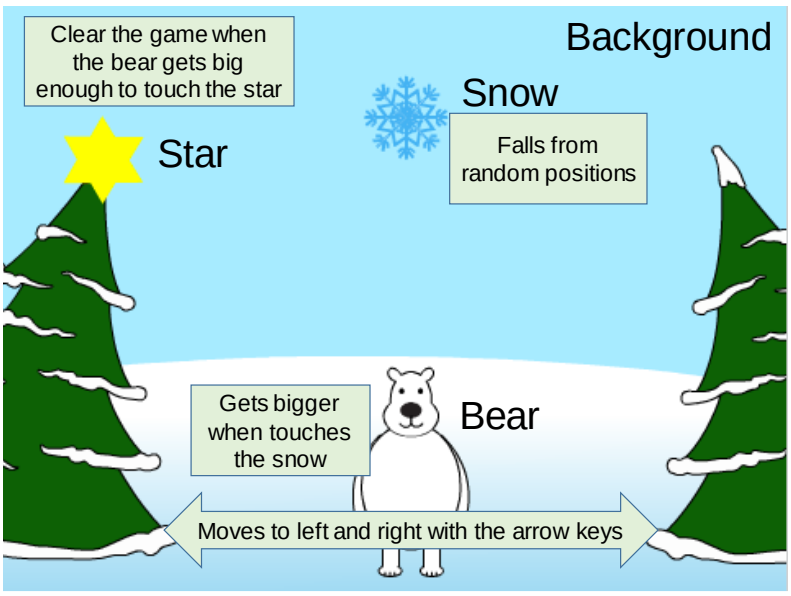


Figure 5. The Snow Game Screen.

We broke down the programming process of this game into small steps and deployed them as a material to be shown in the proposed system running on iPad. Table 1 summarizes the task steps. The game can be created by completing these 11 steps in order.

Table 1

Small Steps of the Task

Step	Task
1	Paste the background and the character
2	Make the bear move to the right
3	Make the bear move to the right only while pressing the right arrow key
4	Make the bear move to the left only while pressing the right arrow key
5	Make the snow fall down
6	Make the snow go up once it falls to the bottom
7	Make the snow fall down from many locations
8	Make the bear get bigger when it touches the snow
9	Make the bear go back to the original size when the system starts the game
10	Make the bear say “game cleared” when it touches the star
11	Make the bear get smaller every 10 seconds

The workshop was held on December 17th, 2017 at Kodomo Mirai-kan located in Kyoto, Japan. Six primary school students (3rd:~2, 4th:~1, 5th:~1, 6th:~2) participated in this workshop. They were given 100 minutes to complete the project.

For this workshop, we made three pairs, each of which was supposed to conduct the mutual evaluation. Participants were paired according to the level of experience of Scratch programming. The characteristics of the pairs are 1) one pair with experienced participants, 2) one pair of beginner participants and 3) one pair with an experienced and a beginner. The participants used the first 30 minutes to learn the basics of programming on Scratch. After that, all participants performed the first two steps together, “step 0” which is a tutorial step to learn how to use the check password of the system and “step 1” which is a task to paste the background and the sprites. From “step 2” each participant followed the given steps. During the programming, university student supporters answered questions from the participants. After all of the participants finished the final step, they filled in the questionnaire.

4.3 Results of the Workshop

4.3.1 Questionnaire Results

Table 2 summaries the content of the questionnaire and its results. There found invalid answers in Q4 and Q7, so they were removed from the results.

Table 2

Questionnaire Contents and Results

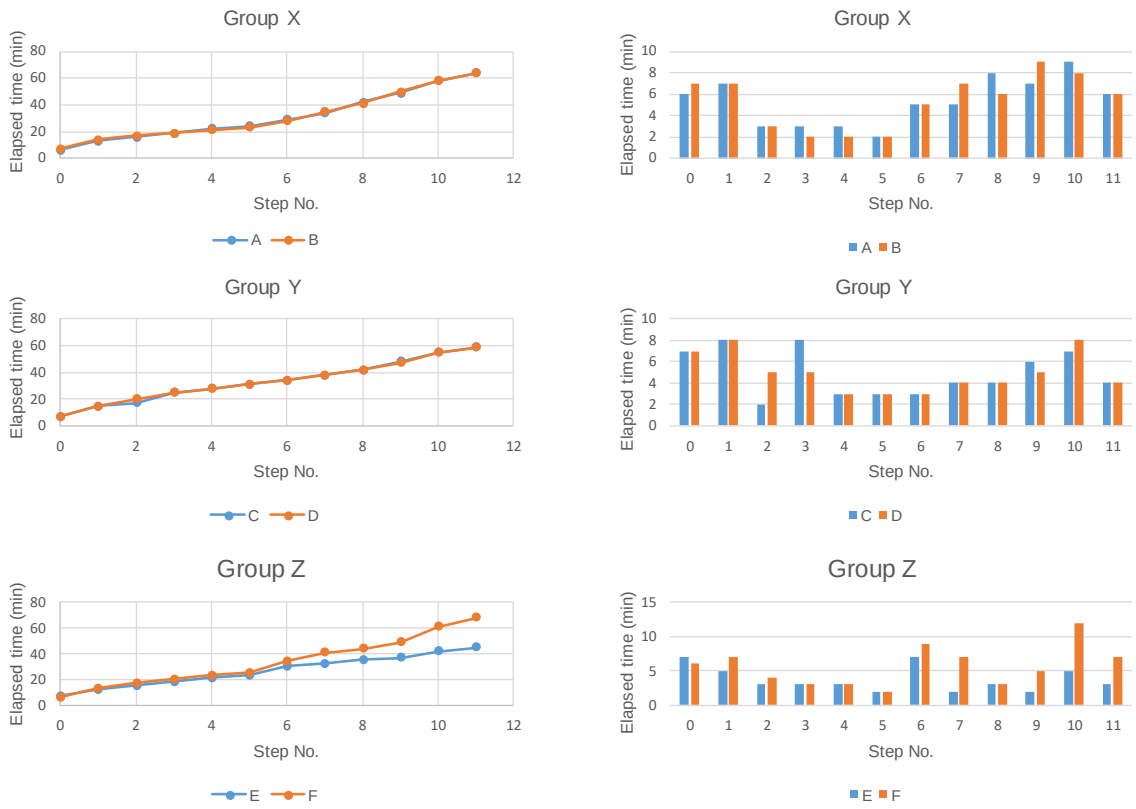
Q1	Have you ever used Scratch?				
	Yes 3	First time 3			
Q2	Could you proceed the tasks smoothly today?				
	Very smooth 5	Smooth 1	Fair 0	Not so smooth 0	Not smooth at all 0
Q3	Please write the reasons. (free writing)				
	• Because university students helped me • Because I have used Scratch before • Because it was fun		• Because it was just for one hour • Because I could ask questions		
Q4	Was it useful to be checked by others?				
	Very useful 3	Useful 1	Fair 1	Not so useful 0	Not useful at all 0
Q5	Please write the reasons. (free writing)				
	• Because I could confirm what I was doing was correct or not • Because I could accurately proceed the tasks • Because making mistakes seems really bad • Because I could ask the person next to me when I came across something I do not know • Because even when I had a confident that I completed the task, if we are working together, we can find the mistake in the program				
Q6	Did you ask someone when you came across something you do not know? (multiple answers allowed)				
	Other participants 3	Supporters 5	Solved by myself 0		
Q7	Was the application easy to use?				
	Very easy 3	Easy 2	Difficult 0	Very difficult 0	
Q8	Do you want to use the application again?				
	Very much 4	Yes 2	No 0	Not at all 0	
Q9	Please write anything about your thoughts on the application. (free writing)				
	• It was a lot of hassle, but I felt great when the “complete” sign appeared on the screen. • It was my first time, but it was good that I could complete. • It was interesting. • I had lots of fun. • I could learn while enjoying.				

Regardless of having an experience in Scratch programming, the satisfaction with programming learning using the proposed system was high (Q2). For the mutual evaluation, most of the participants answered that it was useful for proceeding the tasks (Q4).

4.3.2 Log Results

Figure 6 shows a graph showing the progress of steps (Graph 1) and a graph showing time spent for each step (Graph 2). Six participants are shown as A, B, C, D, E and F. Three pairs are shown as Group X (both experienced), Y (both beginners) and Z (an experienced and a beginner). Because step 0 and 1 are not directly related to programming as mentioned before, we do not value them as a result.

From the results of Graph 1, A and B in Group X had a similar progress of tasks. The same is true for C and D in Group Y. On the other hand, E and F in Group Z had a different progress of tasks. From the results of Graph 2, most of the steps were completed within five minutes. The longest completion time was less than ten minutes. The average time spent on one step, excluding step 0 and 1, was 4.6 minutes.



Graph 1: Progress of steps

Graph 2: Time spent for each step

Figure 6. Results of Logs.

4.3.3 Observation Results

Two pairs were doing programming while talking a lot. On the other hand, one pair was not talking much except while performing the check. The two pairs who talked a lot in pairs were staying close to teach each other as can be seen in Figure 6. They also could find out why the program did not work well. Every time when they complete a step, they were having conversation to share their enjoyment.

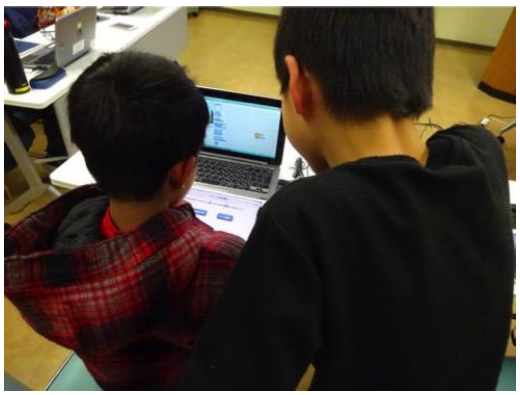


Figure 6. Teaching Each Other.

4.4 Discussion

We observed that children shared their enjoyment in pairs when they completed the steps. The two pairs (Group X and Y) which talked a lot in pairs had almost the same progress of tasks regardless of the individual. From this result, the system could have an effect on participants' feeling of the enjoyment in cooperating and teaching each other.

The pair (Group Z) with few conversations except while performing a check had a different progress depending on the individual. We think that this is because this pair was a group of an experienced and a beginner of Scratch programming. The participants' personality and compatibility might also affect on the conversation frequency.

Even though Group Z did not have a lot of conversation, both participants answered that checking had a positive effect. From this fact, we can say that being recognized objectively by others will lead to a greater sense of accomplishment rather than just completing a task by self-evaluation.

As most of the participants felt that they could proceed the tasks smoothly, we assume that it was effective to present the tasks in small steps. The sense of accomplishment and the time to be spent in one step (4.6 minutes in average) could also lead to participants' feeling of making a smooth progress.

5. Conclusion

In this research, we proposed a programming learning support system that allows children to develop "computational thinking" skills while involving and communicating with others by working on a project broken down into small steps. During the practical implementation at the workshop, we observed that the children did programming while they were feeling an enjoyment and teaching each other. Regardless of having an experience in Scratch programming, all participants were able to proceed the tasks.

Our future work includes the development of a variety of projects to be used in this system. We also have to work on building an authoring tool for creating a project with small steps because our current system requires us to rewrite the application code for a new project to be deployed. The continuous practical implementation and evaluation are also essential to verify the effect of repetitive use of the system.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 16H02925.

References

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc..

Resnick, M., & Robinson, K. (2017). Lifelong kindergarten: Cultivating creativity through projects, passion, peers, and play. MIT press.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.

Falkner, K., Vivian, R., & Falkner, N. (2014, January). The Australian digital technologies curriculum: challenge and opportunity. In Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148 (pp. 3-12). Australian Computer Society, Inc..

Department for Education. (2013). National curriculum for England: Computing programme of study.. London, England: Department for Education.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational researcher, 42(1), 38-43.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. B. (2009). Scratch: Programming for all. Commun. Acm, 52(11), 60-67.

Clark, R. C., Nguyen, F., & Sweller, J. (2011). Efficiency in learning: Evidence-based guidelines to manage cognitive load. John Wiley & Sons.

Atkinson, R. K., Derry, S. J., Renkl, A., & Wortham, D. (2000). Learning from examples: Instructional principles from the worked examples research. Review of educational research, 70(2), 181-214.

WORKSHOP 12 - The 12th Workshop on Technology-enhanced Learning by Posing/Solving Problems/Questions - Analysis and Design of Problems/Questions