Textual vs. Visual Programming Languages in Programming Education for Primary Schoolchildren

Hidekuni Tsukamoto, Yasuhiro Takemura
Dept. of Character Creative Arts
Osaka University of Arts
Osaka, Japan
{hide1123, yasuhi-t }@osaka-geidai.ac.jp

Yasumasa Oomori
College of Education
Joetsu University of Education
Joetsu, Japan
oomori@juen.ac.jp

Isamu Ikeda
Institute of Information Education Support
Kita-Kyushu, Japan
ikeda2015@pic.bbiq.jp

Hideo Nagumo
Dept. of Social Welfare and Psychology
Niigata Seiryo University
Niigata, Japan
nagumo@n-seiryo.ac.jp

Akito Monden
Graduate School of Natural Science and Technology
Okayama University
Okayama, Japan
monden@okayama-u.ac.jp

Ken-ichi Matsumoto
Graduate School of Information Science
Nara Institute of Science and Technology
Nara, Japan
matumoto@is.naist.jp

***Abstract*— The purpose of this research is to compare textual programming languages and visual programming languages from the aspect of motivation. As a textual programming language, Processing programming language was used, and as visual programming languages, Scratch, a derivation of Scratch, Teaching materials offered by code.org, and LEGO Mindstorms EV3 were used. Teaching materials using the textual programming language, and those using the visual programming languages were developed separately. A trial experiment of programming education with the textual programming language was conducted to a cohort of seven primary schoolchildren. Trial experiments with the visual programming languages were conducted twice. In each of them, a cohort of eight primary schoolchildren participated. The motivation of the children was assessed using the questionnaire based on the ARCS (*Attention*, *Relevance*, *Confidence*, and *Satisfaction*) motivation model. The results with the visual programming languages suggested that the motivation scores of the children increased as the class progressed when visual programming languages were used. On the other hand, the results with Processing suggested that the variance of Satisfaction factor increased as the class progressed when textual programming languages were used, which further suggested that the Satisfaction scores of the children spread as the class progressed when textual programming languages were used.**

Keywords—programming education; primary education; textual programming language; visual programming language

INTRODUCTION

Many countries are promoting programming education for young schoolchildren in recent years, and some countries have already adopted programming as a formal subject in primary education [1, 2]. In those countries, almost always visual programming languages (VPLs) are used to teach programming to the children. However, it would be more practical if textual programming languages (TPLs) similar to the ones used for developing real application could be used. Therefore, it is important to investigate the characteristics of TPLs and VPLs as the programming languages used in the programming education for young children. Especially, it would be desirable to obtain the knowledge as to when to start teaching TPLs and the appropriate condition of children to start learning TPLs.

From these considerations, the authors decided to conduct a comparative study between TPLs and VPLs. As the metric for comparing the two groups of programming languages in this research, motivation is employed. Motivation of learners in programming education is considered to be very important. One reason for that is many advocates of VPLs for education purposes expect that VPLs will raise self-efficacy and motivation of the learners [3, 4]. Another reason is that with high motivation, it is possible to keep the dropout rate low, and to make the learners continue learning programming [5].

There are many so called Educational Programming Languages (EPLs), which have been developed for educational purposes. Examples of those VPLs include Scratch, Kodu, Squeak Etoys, and LEGO Mindstorms NXT-G. They are not only simple, but also provide rich and attractive visual outcomes [6]. In fact, most of them are VPLs. VPLs are very popular in programming education for young children like primary schoolchildren and secondary schoolchildren, because they do not require knowledge of programming syntax and provide an environment where compile-time errors are nonexistent [2]. There is also a report saying that VPLs can form long-term opinions about the enjoyment of programming

in the mind of children compared with conventional programming languages [7]. The EPLs used in this research are Scratch browser version, a derivation of Scratch called Pyonkee, teaching materials offered by code.org, and LEGO EV3.

However, there are also some TPLs which have been developed for educational purposes, and have easy to understand and entertaining environments. Microsoft Small Basic, Processing, Python and KidsRuby are examples of those TPLs. If TPLs are used for educational purposes, it is expected that the students would be able to move on to the professional programming languages smoothly later on. In this comparative research, Processing programming language was used as a TPL because it has a longer history as an EPL than other TPLs for beginners, and we can find many reference books and many tutorials on the Web.

When using VPLs, it is relatively easy for the children to make simple games because, there are usually built-in materials and libraries for developing simple games in the language environments. On the other hand, when using TPLs, it is necessary to start from preparing graphics in order to create games. Therefore, in this research, different teaching materials were developed for the children who learn programming with the VPLs and those who learn programming with the TPL. The trial experiment with the VPLs and that with the TPL were conducted to different groups of children. The trial experiment of programming education with the TPL was conducted to a cohort of seven primary schoolchildren. Trial experiments with the VPLs were conducted twice. In each of them, a cohort of eight primary schoolchildren participated.

In order to assess the motivation of the children in the trial experiments, the authors' original questionnaire based on the ARCS (*Attention*, *Relevance*, *Confidence*, and *Satisfaction*) motivation model [8-10] was used. The results with the VPLs suggested that the motivation scores of the children increased as the class progressed when VPLs were used. On the other hand, the results with the TPL suggested that the variance of *Satisfaction* factor increased as the class progressed when TPLs were used, which further suggested that the *Satisfaction* scores of the children spread as the class progressed when TPLs were used.

This paper is organized as following. The next section (section II), contains some comparative research between VPLs and TPLs, and the opinions about the transition from VPLs to TPLs in the related works. In section III and IV, the trial experiment of programming education for primary schoolchildren with VPLs and TPLs are discussed respectively. In section V, the data collection method in this research is presented. In section VI, the analysis of the comparative study is presented. Finally, in section VII, conclusions of this paper are presented.

I. Related Works

Some researchers have conducted a comparative study of VPLs and TPLs. In the paper [7], it was reported that the primary schoolchildren who used Scratch (VPL) continued to use it more than those who used a conventional TPL. However, it is desirable to see the results when more attractive TPLs such as Processing were used. Also, the paper [11] drew a comparison between visual and textual programming environments for Arduino. The authors found that the participants of the experiment felt more confident modifying a program than creating one when using a visual environment, but they felt more confident creating a program than modifying an existing one when using a textual environment. Though the finding is important, unfortunately, it was drawn from adult participants, not primary school children.

Though this is not a comparative study, Touretzky et al. [12] described a three-stage method of teaching programming to children beginning with Kodu, and progressing to Allice and LEGO Mindstorms NXT-G. The idea behind this three stage method was that though highly introductory programming environment offers novices a smooth path to early success in computing, their limited expressiveness must inevitably lead to their abandonment in favor of more powerful conventional languages. They conducted a pilot study with these VPLs in a five-day computer camp for 31 children aged 10-17. They found that at least for older students, it could be better if they used more powerful programming formalism.

The UK government is promoting computer science education for school children. In their new computing curriculum, the children aged between 11 and 14 are required to use two or more programming languages – "at least one of which is textual" – to create their own programs [13].

Maloney et al. [14] discussed the difference between VPLs and TPLs. They are in favor of VPLs, and present examples of points in which TPLs are inferior to one kind of VPLs (Scratch). They argue that variables are invisible, abstract, and difficult to understand in TPLs while they are concrete objects that the user can see and manipulate in Scratch. They also argue that it is easy to get errors by omitting or misplacing the closing delimiters for control structures, while in Scratch a control structure block is an indivisible unit and the closing arm of a loop cannot be misplaced. Though Maloney et al. are in favor of VPLs, they also discuss the weak points of VPLs. They argue that in Scratch it is difficult to increase the number of command blocks because of space limitation while that is not a problem in TPLs.

Malan et al. [15] viewed Scratch as a gateway to TPLs like Java. They argued that most TPLs are not friendly to the students who learn programming for the first time in that the students must master programmatic overhead before programming itself. To validate their proposal, they deployed Scratch for the first time in higher education before they taught the students Java. As their goal was not to improve scores but instead to improve first-time programmers' experiences, they employed subjective measures for their assessment. They concluded that 76% of the students felt that Scratch was a positive influence.

Sengupta et al. [16] introduced students in K-12 science classrooms to visual programming as a pathway to text-based programming. They argue that though VPLs lower the barrier for entry into programming, the learners will find the drag-and-drop nature of visual programming inauthentic if they intend to pursue careers in computing. They propose that to make it easier for the learners to transition from visual programming to

text-based programming, the transition from visual to text-based programming can be achieved by using text-based programming to create visual programming blocks.

II. Programming Education with VPLs

In Japan, VPLs are almost always employed in programming education for young schoolchildren. The authors too employ VPSs such as Scratch and LEGO Mindstorms in summer, winter, and weekend programming classes for primary schoolchildren. For this research, the data collected in the summer classes and winter classes are used. The summer class used two consecutive days (four hours each day), and the winter class used eight days (one hour each day). It may sound irregular to have two different classes, one with two consecutive days, and other with eight days, in two different seasons. The reason for this irregularity is that it is not easy to schedule weekend or after school programming class for schoolchildren in such a way that the class does not coincide with school activities. Also, it is not easy to assemble schoolchildren in the programming classes in such a way that they can attend all the sessions in the class. These are also the reason why we had only a small number of schoolchildren (less than 10) in the programming classes.

The curriculum and the teaching materials for teaching programming to the primary schoolchildren were determined from the experiences of the authors. The curriculum for the summer class with VPLs is shown in TABLE I. There were eight children (two 4th grade boys, three 5th grade boys, and three 6th grade boys) in the class. The classes were conducted in the afternoon of two consecutive days. Each afternoon was divided into four fifty-minute sessions.

In the first session of the first day, the children learned things that are necessary to operate personal computers such as how to use keyboards and mouses, how to cut, copy and paste strings. Most of them had experienced operation of personal computers, but it was necessary to make sure that all participants were able to operate computers. After that, basic operations of Scratch were introduced to the children. The basic operations include drawing lines with changing directions and using repetition blocks. In the second session, Scratch Cards were introduces to the children, and they were used by the children to learn new Scratch code. In the sessions from the third session on the first day to the first session in the second day, the children produced a game. Here, all the participants produced the same game which looked like the one in Fig. 1.

TABLE I. Curriculum of the Summer Programming Classes with VPL

Date	Time	Contents
7/30	13:00-13:50	Basics of programming
	14:00-14:50	Practice using Scratch cards
	15:00-15:50	Game production 1
	16:00-16:50	Game production 2
7/31	13:00-13:50	Game production 3
	14:00-14:50	Game refinement 1
	15:00-15:50	Game refinement 2
	16:00-16:50	Presentation

Fig. 1 The game the children were supposed to produce with Scratch

At first each child made a parrot that fluttered his wings. And then the children learned how to move the parrot up and down in the Scratch stage. After that the children learned how to detect collision and to delete objects. Finally, they learned how to shoot arrows. In the second and third session, the children were told to refine the game in their own way. Here, some suggestions of refinement such as displaying scores, showing "Game Over" message, and making it possible to use a game controller device were given to the children. In the last session on the second day, the children made a presentation of the games they refined, or they made from scratch. In this presentation, each child presented their names, game titles, brief explanation of the game, the points where they exercised their ingenuity, how to play, and the remarks about this programming class. The curriculum for the winter class with VPLs is shown in TABLE II. There were also eight children (three 3rd grade boys, three 4th grade girls, and two 4th grade

TABLE II Curriculum of the Winter Programming Classes with VPL

Date	Time	Contents
11/25	16:00-17:00	Algorithm, Program, and Serial processing with Unplugged Graph Paper Programming
12/2	16:00-17:00	Serial processing with Lightbot
12/9	16:00-17:00	Conditional branch with Unplugged Conditional with Cards, and repetition with Unplugged Getting Loopy
12/16	16:00-17:00	Serial processing and repetitive processing with Code Studio
1/6	16:00-17:00	Repetitive processing and conditional branch with Code Studio
1/13	16:00-17:00	Game production 1 with Pyonkee
1/20	16:00-17:00	Game production 2 with Pyonkee
1/27	16:00-17:00	Serial processing programming with LEGO EV3

boys) in the winter class. In this programming class, the curriculum was designed in such a way that the children learn the concepts of serial processing, repetitive processing, and conditional branching which are the basics of algorithm in the form of a game. The children learned programming by accumulating the experiences of different activities such as programming with pseudo-languages and programming with Scratch and LEGO Mindstorms EV3.

In the first session, the children learned Algorithm, Program, and serial processing using an unplugged teaching material called Graph Paper Programming offered by code.org. In the second session, the children learned serial processing again using an iPad application called Lightbot. In the third session, the children learned conditional branch using an unplugged teaching material called Conditionals with Cards. Also, they learned repetitive processing using an unplugged teaching material called Getting Loopy. In the fourth session, the children learned serial processing and repetitive processing using Course 2, Stage 3 and 6 in Code Studio. In the fifth session, the children learned repetitive processing and conditional branching using Course 2, Stage 6 and 13 in Code Studio. In the sixth session, the children started producing games using a VPL called Pyonkee which is a derivation of Sctatch 1.4. The children programed the games on iPads. In the seventh session, the children continued programming games using Pyonkee. In the eighth session, the children brushed up on conditional branching and repetitive processing using the same unplugged teaching material as the one used in the third session. They also programed LEGO Mindstorms EV3, but without any conditional branching and repetition.

III. Programming Education with TPL

Some researchers who have been using VPLs for teaching programming to primary schoolchildren argued that you need to write many lines of codes just to print "Hello, world" on the screen if you use TPLs. Or, they say that TPLs challenge students to master programmatic overhead before programming itself [15]. Of course, if you use a professional language like Java and professional IDE such as Eclipse to teach programming to programming novices, they would be confused because of the complexity of the environment. However, there are some TPLs such as Processing [17] that do not require such programmatic overhead before programming itself and are suitable for programming education for programming novices.

Authors have been using Processing to teach programming to non-computing major university students, and also to primary schoolchildren. Processing was introduced to programming education for primary schoolchildren, and it was shown that TPLs could be used in programming education for primary schoolchildren [18]. Also, Processing was used in programming education for non-computing major university students and the motivation of the students was analyzed so that the instructors could remediate their teaching materials [19, 20]. Processing is a Java based textual language initiated by C. Reas and B. Fry that facilitate the development of visual projects [17]. The best thing about Processing is that although programming is text-based, the outcomes are not text-based, instead, they are visually attractive artworks. Besides this fact, Processing has many characteristics that make it suitable for being used in programming education for programming novices such as: (1) not requiring creation of Classes, (2) not requiring the codes for creating windows for visual programming, (3) providing many statements for drawing shapes such as ellipses and rectangles, (4) providing a way to check the spelling of reserved words by colors, (5) providing simple ways of making animation. Processing programming environment and a display window are shown in Fig. 2.

The authors conducted a trial holiday class of programming education with a TPL (Processing). There were seven primary schoolchildren (four 5th grade boys, one 6th grade boy, and two 6th grade girls) in the class. Duration of the class was two hours, and the curriculum of the class is shown in TABLE III. In the class, at first the instructor explained what programming is and how a computer operates with a computer program. To help them understand it, the instructor showed a video of projection mapping on some singers dancing on a stage, explaining that programming made possible the projection mapping. After that the instructor explained what Processing was, and taught how to use the Processing development environment. At the beginning of the class, the schoolchildren were told to type the programming codes using keyboards. But after starting the topic "About colors," the schoolchildren were allowed to copy the codes from the text file the instructor provided, and past them in the editor area of the Processing development environment. After pasting, the children were free to change the parameters of the codes.

Instead of just explaining the meaning of the numbers in the brackets of the statements *size*, *point*, *line*, *rect*, *triangle*, and *ellipse*, the instructor asked the children to guess what those numbers do by changing them and seeing the effect of the change. After doing some experiments of changing the numbers, the instructor asked the children to express what they understood from the experiment. This way, the children had the chance to "discover" what they were supposed to know instead of just being taught.

After the short break, the children were taught how to compose colors using three numbers corresponding to the strength of red, that of green, and that of blue. Also, the children were told to observe the computer screen using a compact microscope, and found that each pixel was composed of three primary colors.

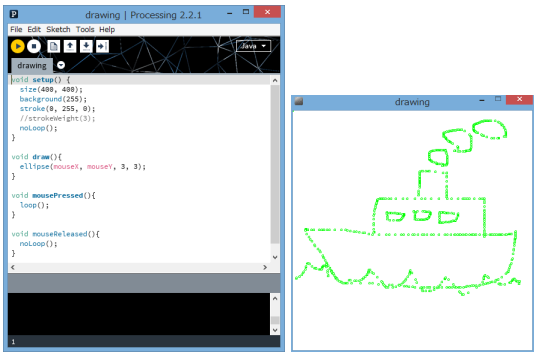

Fig. 2 Processing programming environment and display window

TABLE III CURRICULUM OF THE HOLIDAY PROGRAMMING CLASS WITH TPL

Date	Time	Contents
12/23	13:30-14:20	What is programming?
		Using Processing
		Guess the meaning of the numbers
	14:30-15:30	About colors
		What can we achieve with Processing?
		About programmers

As an example of what could be achieved with Processing, a simple drawing application was introduced to the children so that they could play with it. And at the end, the instructor introduced the programming professions to the children.

IV. DATA COLLECTION

In the past, the authors had been analyzing motivation of the learners in programming education using the authors' original questionnaire based on the ARCS motivation model [18-20]. The ARCS motivation model was introduced by J. M. Keller [8-10]. This model has four factors (*Attention*, *Relevance*, *Confidence*, and *Satisfaction*), and each factor has three lower categories. Therefore there are twelve lower categories in total. The authors developed the questionnaire (ARCS assessment metric), and each question item in the questionnaire was designed to ask if each of the twelve lower categories in the ARCS model was satisfied. Each item was presented using a five-point Likert scale where answer 5 always corresponded to "agree" and answer 1 to "disagree". The lower categories and the corresponding question items of the questionnaire are shown in TABLE IV. Though the question items in the table are in English, actual question items were in Japanese.

The questionnaire based on the ARCS model was conducted after each topic, except at the end of each day in the summer class in 2015 with VPL. Therefore the questionnaire was conducted six times in total in the summer class. Also, it was conducted after each topic in the winter class in 2015 with VPL. Therefore it was conducted eight times in total in the winter class. In the holiday programming class with TPL, the questionnaire was conducted twice. One was after the topic of "Guess the meaning of the numbers", and the other was after the topic of "About programmers".

V. ANALYSIS

The results of the ARCS assessment metric conducted in the three programming classes, summer class with VPL, winter class with VPL, and holiday class with TPL, are shown in Fig. 3, Fig. 4, and Fig. 5 respectively. In each figure, the upper graph shows the mean values of the motivation scores, and the lower graph shows the variances of the motivation scores. In each graph, the legend shows the factors in the ARCS model: *A* stands for *Attention, R* for *Relevance*, *C* for *Confidence*, *S* for *Satisfaction*, and *All* means the total of the four factors. Also, some legends are accompanied with one asterisk (*) or two

TABLE IV THE QUESTION ITEMS IN THE QUESTIONNAIRE

	Attention
A1	Perceptual arousal: Did the programming education give you a pleasant surprise?
A2	Inquiry arousal: Did you feel that you wanted to learn more during the programming education?
A3	Variability: Could you study without getting bored because there were variations in the learning content?
	Relevance
R1	Familiarity: Did you feel that the learning content was familiar?
R2	Goal orientation: Did you understand the goal and the importance of the learning?
R3	Motive matching: Did you have chances to select the learning methods that were suitable to you?
	Confidence
C1	Learning requirements: Is the goal you should reach clear?
C2	Success opportunity: Did you have an occasion to feel that you had written your programs well?
C3	Personal control: Did you feel that you wrote your program well because of your efforts and ability?
	Satisfaction
S1	Natural consequence: Did you have occasions to use your newly acquired knowledge?
S2	Positive consequence: Were you happy when you made a good program?
S3	Equity: Was your accomplishment fairly evaluated with a consistent standard?

asterisks (**). One asterisk on top of the legend indicates that the difference between the value of No.1 survey and that of No. 2 survey is statistically significant ($p < 0.05$), and two asterisks for $p < 0.01$. The asterisks under the legends indicates the same thing except that the difference is between No. 1 survey and the last survey. In the case of winter class with VPL, No.7 survey was considered as the last survey because one child was absent when the No. 8 survey was conducted.

According to Fig. 3 and Fig. 4, it seems that the motivation scores were increased from No. 1 survey towards the last survey. In fact, the scores of all the four factors increased from the first survey to the last survey in the summer class with VPL. In winter class with VPL, Confidence score increased from No. 1 survey towards No. 2 survey, and also from No. 1 survey towards No. 7 survey. The variances of the scores of the two classes with VPL were relatively small when comparing with those of the class with TPL. Therefore it could be said that the motivation of the majority of the children who participated in the classes with VPL to learn programming was raised as the children attended more classes.

On the other hand, according to Fig. 5, the motivation scores of the children who participated in the holiday class with TPL did not show any evidence of increasing. Of course, the results of the two classes with VPL and those of the holiday class with TPL are not directly comparable because the survey was conducted only twice in the class with TPL while more than six surveys were conducted in the classes with VPL. It is noteworthy, however, that the motivation of the children seems to increase constantly without much fluctuation. It is already known that the motivation of students fluctuate as the course progresses depending on the teaching contents and study environments when teaching programming to university

students or older people [21]. Therefore, a steady increase of motivation scores of those children is not that natural.

From these observations, it could be said that, from the standpoint of increasing the motivation of the learners, VPLs such as Scratch and Pyonkee are more suitable to be used for programming education for primary schoolchildren than TPLs such as Processing. Of course, it is not to say that VPLs are always better than TPLs for programming education for primary schoolchildren, because assessing motivation of the children is just one of many evaluation methods. There are many other evaluation tools for programming education [22].

Another observation is that the results with Processing suggested that the variance of Satisfaction factor increased as the class progressed when textual programming languages were used, which further suggested that the Satisfaction scores of the children spread as the class progressed when textual programming languages were used.

Maloney et al. [14] argued that in Scratch it is difficult to increase the number of command blocks because of space limitation while that is not a problem in TPLs. In the Scratch browser version used in this research, this space limitation of command set was not a big problem because all the commands required in the curriculum could be found in the command palettes by scrolling them down a little bit.

The weak point of this research is that the durations and curriculums are different for different programming classes. As mentioned before, it was due to the difficulty of arranging the classes in such a way that the sessions in the classes do not coincide with the school activities for all the participants. Therefore, the comparison is not very sound. This condition

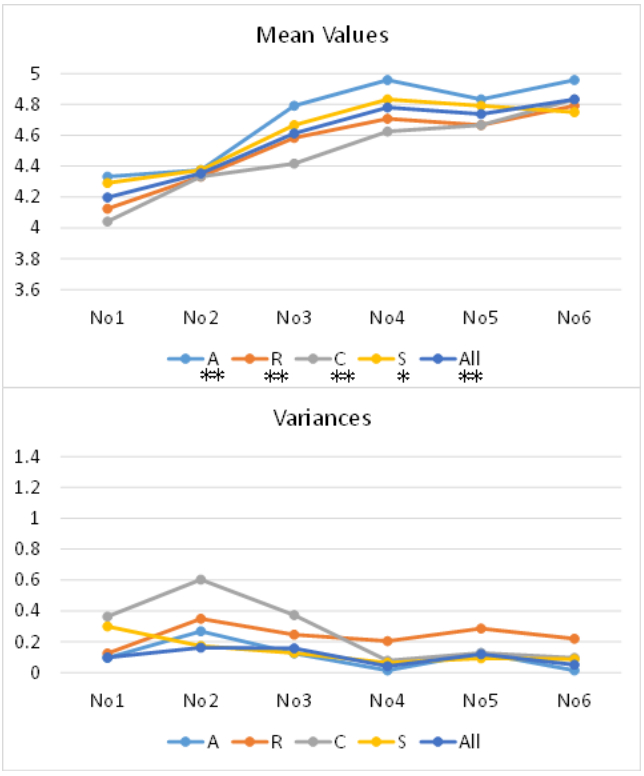

Fig. 3 The scores of the four factors for the summer class with a VPL

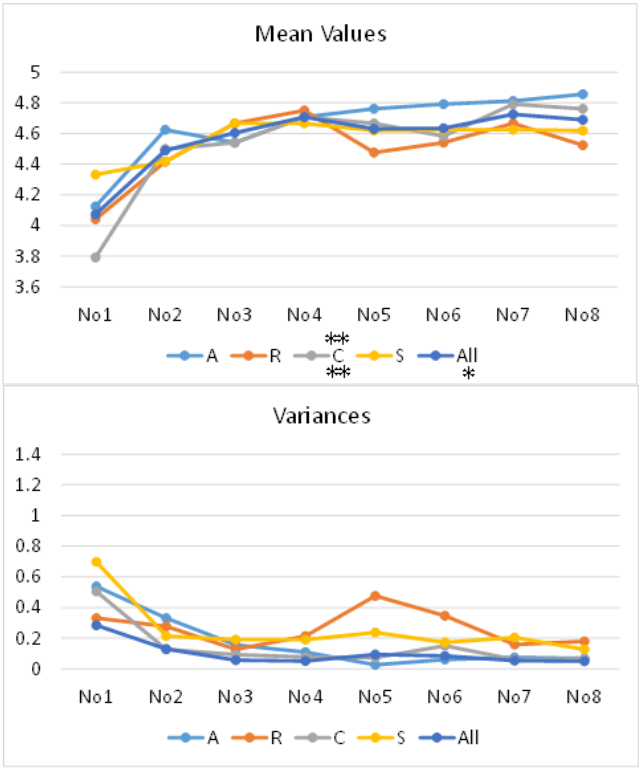

Fig. 4 The scores of the four factors for the winter class with VPLs

would be improved because the Japanese government announced in May 2016 that it would make programming a compulsory subject in primary schools from 2020 [23]. Because of this announcement, we could expect that teachers and parents would be interested in programming education, and would give higher priority to the weekend and after school programming classes.

VI. Conclusions

In this research, the programming education for primary schoolchildren with VPLs and that with TPLs are compared from the view point of motivation of the children. In assessing the motivation of the children, the authors' original questionnaire based on the ARCS motivation model was used. Two programming classes, one in summer and other in winter, with VPLs were conducted and eight primary schoolchildren participated in each class. In the summer class, the survey with the authors' questionnaire was conducted six times, and in the winter class, the survey was conducted eight times. One programming class with a TPL was conducted, and seven primary schoolchildren participated in the class. In the two classes with VPLs, it was observed that the motivation scores of the children increased as the class progressed, while in the class with a TPL, increase of the motivation scores was not observed. Therefore, it could be said that, from the standpoint of increasing the motivation of the learners, VPLs are more suitable to be used for programming education for primary schoolchildren than TPLs. The limitation of the research and its results were also discussed.

The authors felt that motivation alone is not enough to evaluate programming education for primary schoolchildren,

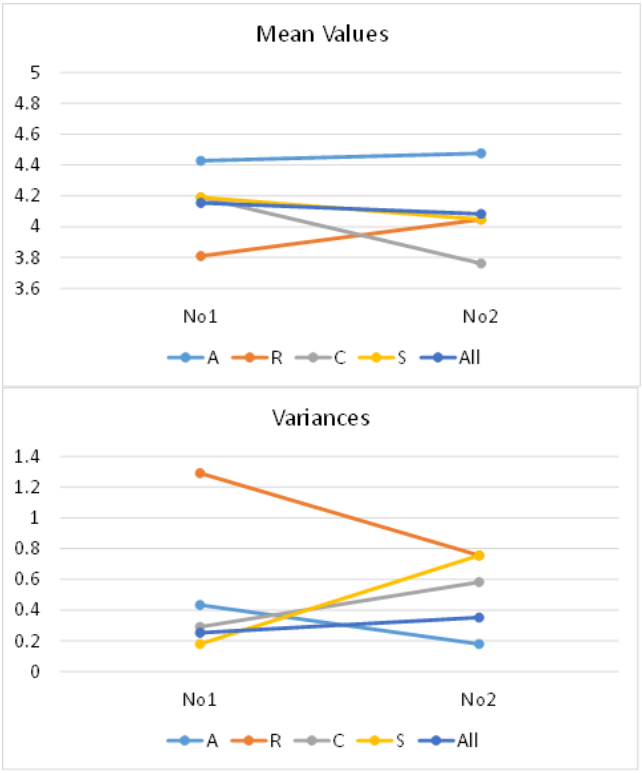

Fig. 5 The scores of the four factors for the holiday class with a TPL

and that if the authors could find positive effects of programming education to other subjects, it would be easier for the author to show the importance of programming education in primary schools in Japan. Therefore, in future research, the authors would like to develop a new metric that can measure the effects of programming education in other subjects.

ACKNOWLEDGMENT

This work was partly supported by Grants-in-Aid for Scientific Research of Scientific Research (C) No.JP16K01087 from Japan Society for the Promotion of Science.

REFERENCES

[1] C. Williams, E. Alafghani, A. Daley Jr., K. Gregory, and M. Rydzewski, "Teaching Programming Concepts to Elementary Students," 2015 Frontiers in Education Conference Proceedings (FIE 2015), 2015, pp. 706-714.

[2] C. Duncan, T. Bell, and S. Tanimoto, "Should Your 8-year-old Learn Coding?," ACM WiPSCE'14, 2014, pp. 60-69.

[3] O. Meerbaum-Salant, M. Armoni, and M. Ben-Ari, "Habits of Programming in Scratch," Proceedings of the 16th Annual Conference on Inovation and Technology in Computer Science Education (ITiCSE' 11), 2011, pp. 168-172.

[4] M. Armoni, O. Meerbaum-Salant, and M. Ben-Ari, "From Scratch to 'Real' Programming," ACM Transactions on Computing Education, Vol. 14, No. 4, 2015, pp. 25:1-25:15.

[5] S. W. Martins, A. J. Mendes, and A. D. Figueiredo, "A Strategy to Improve Student's Motivation Levels in Programming Courses," 2010 Frontiers in Education Conference Proceedings (FIE 2010), 2010, pp. F4F-1 – F4F-7.

[6] J. Choi, S. An, and Y. Lee, "Computing Education in Korea – Current Issues and Endeavors," ACM Transactions on Computing Education, Vol. 15, No. 2, 2015, pp. 8:1-8:22.

[7] B. Kaucic and T. Asic, "Improving introductory programming with Scratch?," Proceedings of 34th International Convention MIPRO, 2011, pp. 1095-1100.

[8] J. M. Keller and K. Suzuki, "Use of the ARCS motivation model in courseware design," in D. H. Jonnasen (Ed.), Instructional designs for microcomputer courseware, Lawrence Erlbaum Associates., 1987, pp. 401-434.

[9] R. M. Gagne, W. W. Wager, K. C. Golas, and J. M. Keller, "Principles of Instructional Design," Fifth Edition, Wadsworth Pub Co, 2005. (r)

[10] J. M. Keller, "Motivational Design for Learning and Performance: The ARCS Model Approach," Springer, 2010.

[11] T. Booth, and S. Stumph, "End-User Experiences of Visual and Textual Programming Environments for Arduino," End-User Development, Lecture NOtes in Computer Science Vol. 7897, 2013, pp. 25-39.

[12] D. S. Touretzky, D. Marghitu, S. Ludi, D. Bernstein, and L. Ni, "Accelerating K-12 Computational Thinkig Using Scaffolding, Staging, and Abstraction," SIGCSE'13, 2013, pp. 609-614.

[13] GOV.UK Department for Education, "Statutory guidance National curriculum in England: computing programmes of study," https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study.

[14] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The Scratch Programming Language and Environment," ACM Transactions on Computing Education, Vol. 10, No. 4, 2010, pp. 16.1-16.15.

[15] D. J. Malan and H. H. Leitner, "Scratch for BuddingComputer Scientists," ACM SIGCSE' 07, March 2007, pp. 223-227.

[16] P. Sengupta, A. Dickes, A. V. Farris, A. Karan, D. Martin, And M. Wright, "Programming in K-12 Science Classrooms," Communications of the ACM, Vol. 58, No. 11, 2015, pp. 33-35.

[17] C. Reas and B. Fry, "Processing: A Programming Handbook for Visual Designers and Artists," MIT Press, 2007.

[18] H. Tsukamoto, Y. Takemura, H. Nagumo, I. Ikeda, A. Monden, and K. Matsumoto, "Programming Education for Primary Schoolchildren Using a Textual Programming Language," 2015 Frontiers in Education Conference Proceedings (FIE 2015), 2015, pp. 1008-1014.

[19] H. Tsukamoto, Y. Takemura, H. Nagumo, A. Monden, and K. Matsumoto, "Prediction of the Change of Learners' Motivation in Programming Education for Non-Computing Majors," 2014 Frontiers in Education Conference Proceedings (FIE 2014), 2014, pp. 1421-1427.

[20] H. Tsukamoto, Y. Takemura, H. Nagumo, A. Monden, and K. Matsumoto, "The Effects of Teaching Material Remediation with ARCS-Strategies for Programming Education," 2013 Frontiers in Education Conference Proceedings (FIE 2013), 2013, pp. 717-723.

[21] H. Tsukamoto, H. Nagumo, Y. Takemura, and N. Nitta, "Change of Students' Motivation in an Introductory Programming Course for Non-Computing Major," Proceedings of the 12th IEEE Conference on Advanced Learning Technology (ICALT 2012), 2012, pp. 124-125.

[22] L. Werner, J. Denner, and S. Campe, "Children Programming Games: A Strategy for Measuring Computational Learning," ACM Transactions on Computing Education, Vol. 14, No. 4, 2014, pp. 24.1-24.22.

[23] fossBytes, "Japan Just Made Computer Programming A Compulsory Subject In Its Schools," http://fossbytes.com/japan-computer-programming-compulsory-subject-schools/