Learning System for Computational Thinking using Appealing User Interface with Icon-Based Programming Language on Smartphones

Kazunori Sakamoto
National Institute of Informatics
Tokyo, Japan
exkazuu@nii.ac.jp

Koichi Takano
OBIC Co., Ltd.
Tokyo, Japan
furbishbiaskoichi7331306@gmail.com

Hironori Washizaki, Yoshiaki Fukazawa
Waseda University
Tokyo, Japan
washizaki@waseda.jp, fukazawa@waseda.jp

***Abstract*—Computational thinking is one of the most important skills for using computers. Most existing learning environments for computational thinking work only on desktop or laptop computers although the popularity of smartphones has rapidly been growing. Moreover, Most existing programming languages are based on English. However, Japanese students tend to not like such languages due to English.**

We propose a gamified learning system using an appealing user interface with a novel icon-based non-verbal programming language. Our system works on smartphones with which many Japanese teenager students are more familiar than PCs. Our system uses an appealing interface and icons to motivate university students to learn programming through playing. We applied an appealing interface which a female student designs for other female students into our system, and then, conducted an experiment with 16 female students from Waseda University in Tokyo, Japan to evaluate our system. We confirmed our system encouraged and motivated the students to learn programming.

I. Introduction

Computers are widely used and necessary for daily tasks. The achievement of ubiquitous computing will become reality due to the wide use of smartphones [1]. Although the spread of computers gives birth to the necessity of computational thinking [2], investigation results of the ACM and other organizations show the relative unpopularity of computer science compared to more traditional mathematical disciplines [3].

Redmond et al. studied students in computer science at Stanford University [4]. They reported that one of the biggest obstacles to the students taking a computer science (CS) major was starting CS classes too late in their undergraduate education. They also argued that starting computer science early is important.

To motivate students for learning programming, we propose a gamified learning system that is suitable for Japanese university students, using an appealing user interface with a novel icon-based non-verbal programming language. We summarize eight features of our learning environment as follows.

- **Affinity for smartphones** Our system is an Android application working on smartphones instead of desktop or laptop computers. Smartphones are more popular and familiar than PCs with this demographic.
- **Game** Our system provides a game which requires users to write programs in our icon-based language. Users unconsciously learn essences of programming through playing our system.
- **Problem** Our game provides problems called stages that users should solve. Users can easily understand what they should do due to these stages with obvious goals.
- **Appealing user interface** Our system uses appealing graphics for the user interface. Such graphics motivate students to play a game and learn computational thinking.
- **Visualization of behavior** Our system is equipped with a language processor that visualizes an execution of a user program by showing the user avatar dancing and highlighting an executing line.
- **Icon-based non-verbal language** Our language consists of only 11 icon images, which indicate the user avatarfs actions. Users can more easily write programs using icons than text.
- **Native language (Japanese)** The icons can be converted into Japanese so that users can confirm what the icons mean.

Users write code using our language to complete the stages in the game. Users can experience programming through playing our game and writing code. Our system motivates Japanese students to learn programming.

II. Key Designs

We investigated why many Japanese students tend not to want to study computer science and learn programming. We found the following three obstacles: unfamiliarity with PCs, programming languages with difficult English words, and poorly designed user interfaces. To remove such obstacles, we designed our system as follows.

A. Affinity for Smartphones

The popularity of smartphone is growing rapidly. In particular, young people who are between 10 and 19 ages use smartphones rather than PCs in Japan to access the Internet.

TABLE II. UTILIZATION RATE OF SMARTPHONES BY GENDER AND AGE GROUP IN 2012

Gender	Male						Female					
Ages	10-19	20-29	30-39	40-49	50-59	60-	10-19	20-29	30-39	40-49	50-59	60-
Rate of smartphone utilization	51.7	58.9	53.7	44.3	33.8	16.8	47.6	58.5	46.3	33.1	23.6	12.4

TABLE I. TIME SPENT USING SMARTPHONES AND PCS BY GENDER AND AGE GROUP IN 2012

	Smartphone	PC
Male (10 - 19 ages)	121	113
Male (20 - 34 ages)	102	162
Female (10 - 19 ages)	149	101
Female (20 - 34 ages)	121	151

Nielsen Co., Ltd. reported that the time people spent on the Internet with smartphones is longer than with PCs in Japan [5]. The company also reported that the monthly time a person spent on the Internet with smartphones and with PCs in September 2012 was 1,492 and 1,301 minutes on average, respectively.

ASATSU-DK INC. reported that teenagers use smartphones to access the Internet more than other age groups [6]. Table I lists the time people spent on the Internet with smartphones and PCs per day in 2012. The company reported that the time a teenage male spent on the Internet with smartphones and PCs in 2012 per day was 121 and 113 minutes on average, respectively. They also reported that the time a teenage female spent was 149 and 101 minutes on average, respectively.

Impress R&D reported that more young people use smartphones more often than older people [7]. Table II lists the rate of utilization of smartphones by gender and age group. Approximately half the teenagers surveyed had smartphones.

While many people have and use PCs, the above surveys indicate smartphones will be more ubiquitous than PCs in the future. However, writing programs with current programming languages on smartphones is obviously more difficult than on PCs. Therefore, a learning system should work on smartphones rather than PCs and should provide an educational programming language designed for smartphones.

B. Icon-Based Educational Programming Language

Many Japanese students tend not to like programming because most of programming languages are based on English.

Igo et al. conducted questionnaire study on reasons bachelor Japanese students from Yamaguchi University dislike programming [8]. They reported that 69% of 39 students who attended the programming lecture at Yamaguchi University disliked C programming language, and 31% liked it; 46% were intimidated by English, and 54% were not. They reported that 78% of the students who were intimidated by English disliked C programming language and 22% liked it, and 52% of the students who disliked C programming language were intimidated by English and 48% were not.

That is, the students who were intimidated by English tended to dislike C programming language and the students who hated C programming language tended to be intimidated by English. This result indicates that one of the reasons students dislike C programming language is because C programming is based on English. Therefore, a learning environment for Japanese students should avoid using English.

C. Appealing User Interface

A user interface is one of the most important things to motivate users to use the software. Cho et al. reported that good user-interface designs could motivate learners to continue using their e-learning tools [9]. Nittono et al. found that viewing such cute (or Kawaii in Japanese) images promoted careful behavior and narrow attentional focus regardless of gender [10].

That is, an appealing user interface (e.g. using Kawaii images) can motivate students to learn programming and help students concentrate on programming. However, existing learning environments have no such user interface. Therefore, a learning environment for students should employ such an appealing user interface.

III. OVERVIEW OF OUR SYSTEM

We developed our system with an icon-based non-verbal programming language. Users learn programming based on three computational thinking concepts; *sequences*, *loops*, and *parallelisms*, through playing a game. The game consists of 13 stages which are similar to problems of puzzle games.

Each stage shows an example dancing avatar (chicken). Users write a set of commands which replicates the dance steps. The userfs avatar (baby chick) and the example avatar (chicken) dance the same when the userfs dance steps correctly replicate the examplefs. Our system then shows the user that the dance steps are correct.

Figure 1 shows two game screenshots in which a user successfully (right) and unsuccessfully (left) solved a stage. The left screenshot is wrong because the example avatar raised the left wing while the user avatar raised the right wing. The avatars were designed by a female university student for other female university students. Note that we can replace these images with other images designed for others.

Our system was developed as an Android application; therefore, users can learn computational thinking concepts with our system anytime and anywhere. Moreover, people communicate with others via smartphones more frequently than PCs in Japan. Thus, smartphones will be more familiar and deep for people than PCs. This tendency does not apply to only Japan but also worldwide.

IV. PROGRAMMING LANGUAGE DESIGN

We designed our programming language to address the following issues for Japanese students.

- Most programming languages are based on characters and it is assumed that users write code not with smartphones but with PCs. People who frequently use smartphones rather than PCs tend to believe programming is not easy.

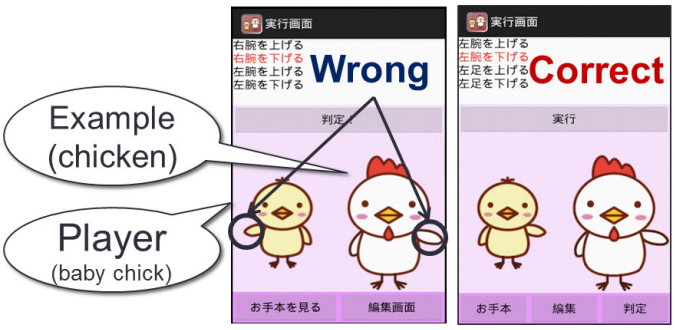

Fig. 1. Screenshot of our Android application (wrong and correct answers on left and right sides, respectively)

- Most programming languages force users to write code containing English words, symbols, and numbers. People who dislike English or mathematics tend to believe that programming is difficult and boring.

Figure 2 shows the Backus-Naur Form (BNF) of our language. Our language consists of nine action commands for making the user avatar dance and one loop command for representing repetition behavior. Eight of the nine action commands indicate raising or lowering the user avatarfs left wing, right wing, left leg, or right leg. The last action command indicates jumping. The loop command indicates repeated action commands with the specified numbers. Our language use only numbers for indicating the number of repetitions.

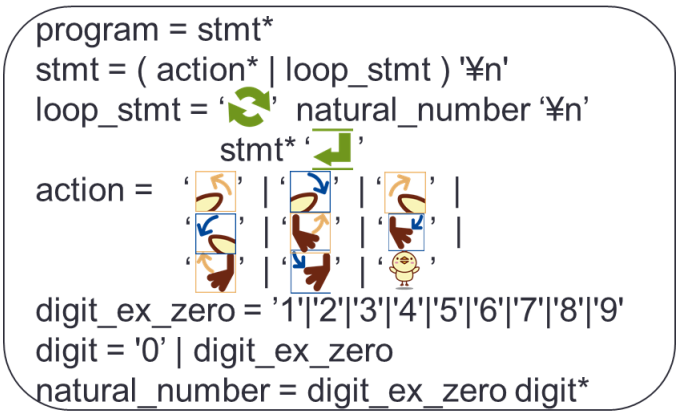

Fig. 2. Backus-Naur Form of our icon-based programming language

The images in the BNF are icons similar to emoticons that are commonly used in e-mail in Japan. The nine action commands, the loop command, and ending sing of loops can be represented by Japanese statements or icons, which are interconvertible. That is, users can write programs with only icons and numbers for loops without character-based statements. Figure 3 shows same program written using character-based statements (Japanese) and numbers and using icons and numbers. Users can convert programs by changing the upper tabs, as shown in Figure 3.

The eight action commands, except jumping, indicate changes in the state of the user avatar. For example, after executing the raising right wing command, the user avatar continues to raise the right wing until executing the lowering

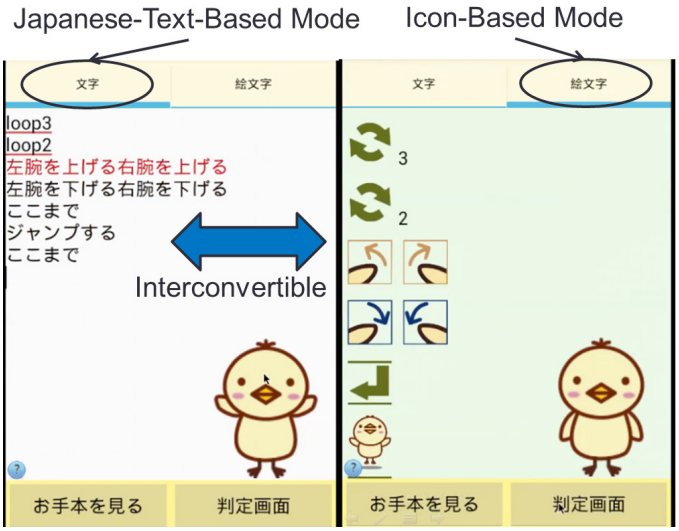

Fig. 3. Same program written using Japanese and numbers (on left side) and using icons and numbers (on right side)

right wing command. On the other hand, the jumping command indicates just that command, i.e., after executing the jump command, the user avatar will automatically land without executing any other command.

The `stmt` indicates each line in the program written in our language. It can contain multiple `actions`, which are the action commands, or one `loop_stmt`, which is the loop command. Our system executes each line of the program sequentially from the top line to the bottom line at a constant time interval. Note that our system skips lines which contain no action command. The `stmt` is executed at the same time, that is, multiple action commands on the same line are executed simultaneously.

For example, the third line of the program in Figure 3 contains the raising left wing and the raising right wing commands. When executing the third line, the user avatar raises both wings simultaneously. The user avatar will lower both wings at the next time because the fourth line contains the lowering left wing and the lowering right wing commands. When executing the sixth line containing only the jumping command, the user avatar jumps and will land at the next timing.

Our system conducts syntax and semantic analyses similarly to other programming languages. Our system parses programs written in our language and reports syntax errors such as the lack of ending signs of the loop commands and wrong Japanese statements to users in the syntax analysis. Our system also analyzes and executes programs and reports semantic errors due to impossible actions such as raising both legs during execution to users in the semantic analysis. When semantic errors occur, the user avatar falls down. Our system provides an appealing user interface even if errors occur. Moreover, to make the program understandable, our system highlights the current execution line with red when executing the program.

There are four reasons our system uses icons instead of words. Icons are 1) familiar to smartphone users, 2) informative, 3) appealing, and 4) language-independent.

1) In Japan, most of people use icons to write e-mails with mobile phones and smartphones. Thus, icons are familiar with smartphones users. 2, 3) Icons contain more information than characters. Users can write programs with fewer icons than characters. Moreover, icons can be designed to be more appealing than characters. The icons were also designed by a female university student for other female university students. Note that we can replace these icons with other icons designed for others.

4) Icons do not depend on characters so users can learn computational thinking concepts independently from native languages. If users do not want to use Japanese, they can write programs using icons with our language. Our system supports the inter-conversion between Japanese statements and icons to just explain what icons mean. Moreover, we can replace Japanese statements with other statements in other languages.

V. Computational Thinking Concepts

Users can learn three computational thinking concepts: *sequences*, *simultaneity*, and *loops*. We now explain how these concepts are learned. We provide 13 stages; 3 stages for learning sequences, 3 sages for learning simultaneity, 4 stages for learning loops, and 3 stages for revising the three concepts.

A. Sequences

In general, programs are executed in a certain order. Most procedural programming languages execute programs from the top line to the bottom line. Our system executes each line sequentially from the top line to the bottom line at a constant time interval. For example, when the third line in Figure 3 is executed, the next line will be executed after the expiration of that time interval.

The first stage requires users to write programs where the user avatar 1) raises the left wing, 2) lowers the left wing, 3) raises the right wing and 4) lowers the right wing. Users will notice that there is a rule about the sequence of executing commands through playing this stage. Therefore, users learn *sequences*.

B. Simultaneity

In computer science, parallelism is one of the most important concepts. Although simultaneity is different from parallelism, we should learn about simultaneity and distinguish simultaneity from sequences to understand parallelism. Our system executes multiple action commands simultaneously on the same line. For example, when the third line in Figure 3 is executed, the user avatar raises both left and right wings simultaneously.

The fifth stage requires users to write programs where the user avatar 1) raises the left wing and the right leg, 2) lowers the left wing and the right leg, 3) raises the right wing and the left leg, and 4) lowers the right wing and the left leg. Users will notice that there is a rule about simultaneously executing commands through playing this stage. Therefore, users learn simultaneity.

C. Loops

Loops are also one of the most important concepts, and most programs contain loops. Our system supports only repetition with the specified number of iterations because such repetition is the most basic loop. Our system repeatedly executes commands, which are surrounded with a loop command and an ending symbol. For example, the third and fourth lines in Figure 3 are executed six times and the sixth line is executed three times because there are two loops with three and two iterations.

The seventh stage requires users to write programs in which the user avatar repeats the following four action commands three times: 1) raises the left leg, 2) lowers the left leg, 3) raises the right leg and 4) lowers the right leg. Users will learn that a loop command is useful for writing repeated action commands through playing this stage. Therefore, users learn loops.

VI. Evaluation

We conducted an experiment for evaluating the following research questions (RQs). Although our system does not distinguish users by gender and native language, we investigate the following RQs for female university students because the user interface of our system is designed by a Japanese female university student for other Japanese female university students.

- **RQ1:** Does our learning environment improve Japanese female university studentsf impressions of programming?
- **RQ2:** Does our learning environment motivate Japanese female university students to learn programming?
- **RQ3:** Is our learning environment more effective for learning three computational thinking concepts than by reading lecture notes?
- **RQ4:** Do Japanese female university students have a good impression of our learning environment?

Sixteen Japanese female university students from Waseda University in Tokyo, Japan participated in our experiment. Nine out of the 16 were liberal arts students and 7 were science students. We divided the 16 students into a group that used our learning environment and a group that did not. Note that we divided the liberal arts students and science students into the two groups as evenly as possible.

We conducted the following steps of our experiment for the group that used our learning environment.

1) We gave a preliminary questionnaire to determine the studentsf impressions on programming and their motivations for learning programming.
2) The students used our learning environment for approximately 40 minutes.
3) We conducted an achievement test for determining the depth of understanding of the three computational thinking concepts of *sequences*, *simultaneity*, and *loops*.
4) We gave a post-investigation questionnaire with the same questions as the preliminary one.

The group that did not use our learning environment learned computational thinking concepts with lecture notes for approximately 40 minutes instead of the third step.

To investigate RQ1 and RQ2, we gave preliminary and post-investigation questionnaires to eight of the students who used our learning environment. We asked the following four questions about their impressions of programming. A) Do you feel programming is mainly geared toward males? B) Do you feel programming is scientific and not suitable for liberal arts students? C) Do you feel programming is difficult? D) Do you feel programming is boring? We also asked the following question about their motivations for learning programming. E) Do you not want to learn programming? The students answered our questions based on a six-point scale (1 strongly agree and 6 strongly disagree).

Figure 4 is a box plot of the questionnaires results. The result indicates that the studentsf impressions of programming and motivations for learning programming improved because the positions of all the boxes moved upwards compared with the preliminary results. Moreover, all the highest values and the central values, except for the question A, of the answers for our post-investigation questionnaires improved in comparison with them of our preliminary investigation questionnaires. On the other hand, all the lowest points of the answers did not change between our preliminary and post-investigation questionnaires. Thus, our learning system can generally improve Japanese female university students' impressions of programming, but it is difficult to mitigate very bad preliminary impressions of programming.

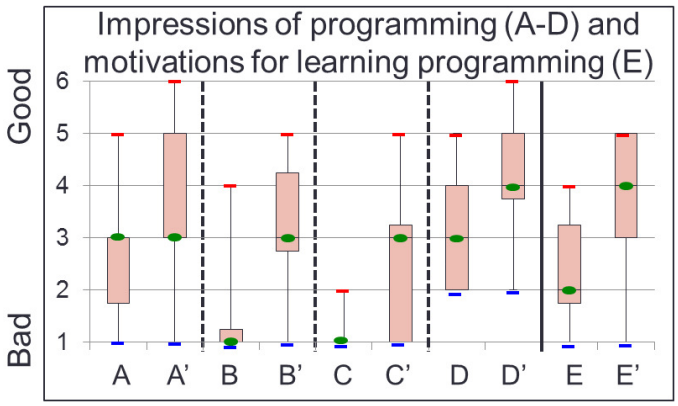

Fig. 4. Questionnaire results on impressions of programming (A-D) and on motivations for learning programming (E). Note that A-E and Af-Ef indicate preliminary and post-investigation questionnaires.

To investigate RQ3, we gave an achievement test to all 16 students. Our achievement test consists of three problems for determining how effectively they understood the three computational thinking concepts of sequences, simultaneity, and loops. We used a different game where the avatarfs movements were two-dimensional instead of the dancing for fairness. We also provided a programming language similar to our dance language in our learning system for our achievement test. The language consisted of four action commands; left, right, up, and down on a one-hundred-square board instead of nine action commands. The language also had the same loop command as the one in our dance language.

The first and second problems required the students to write the track of the avatar which moves on the board with the given programs with loop commands. The programs of the first and second problems used one loop command and dual loop commands, respectively. The third problem required the students to write a program that moved the avatar in a zigzag line with sequential commands. The fourth problem required the students to write a program that moved the avatar diagonally with simultaneous commands.

Figure 5 shows the results of our achievement test. All the students that used our system solved the first, second, and third problems correctly. On the other hand, several students that did not use our system did not solve these problems. Moreover, more students that used our system solved the fourth problems correctly than those who did not use it. Thus, our system helped them learn the three computational thinking concepts more effectively compared with using lecture notes.

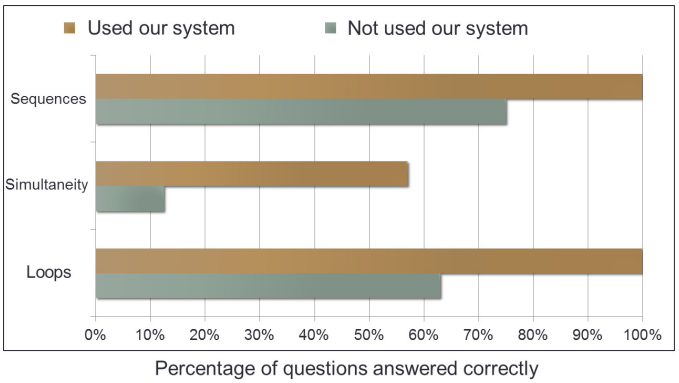

Fig. 5. Percentages of questions answered correctly in achievement test

To investigate RQ4, we gave a questionnaire to the eight students who used our system to determine their impressions of the system. We asked the following four questions. F) Do you feel our system is fun? G) Do you feel our system is easy to play? H) Do you feel you want to introduce our system to your friends? I) Do you feel you want to use our system more? The students answered our questions based on a six-point scale (1 strong disagree and 6 strongly agree).

Figure 6 shows our questionnaire results. All the students except one said our system was fun, easy, and attractive. Thus, our system may be attractive to Japanese female university students so that our system is very useful to teach the fun and the attractive of programming. Only one student said our system reminded her of a bad and boring programming lecture she previously attended. This impression relates to the questionnaire about impressions of programming and motivations for learning programming and indicates that it is difficult to improve impressions of Japanese female university students who believe programming is boring and who are not skilled in programming.

VII. Related Work

There have been studies on how to encourage young girls to learn programming and computational thinking [11], [12]. Scratch is a programming environment that enables young people to create their own interactive stories, games, and

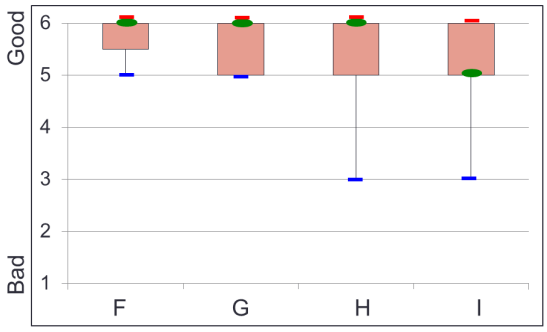

Fig. 6. Results of questionnaire on impressions of our system

simulations [11]. The core users are between the ages of eight and 16. Young people can learn seven computational thinking concepts: *sequences*, *loops*, *parallelism*, *events*, *conditionals*, *operators*, and *data*. CodeSpells is a video game that enables young people to learn Java programming [12]. CodeSpells successfully taught 40 girls between the ages of 10 and 12 Java programming for the first time in their experiment.

Although the effectiveness of Scratch and CodeSpells is confirmed in their experiments, their environments work on only PCs. However, our system works on smartphones, thus, it is more familiar with smartphone users who are increasing.

There also have been studies on learning environments which work on smartphones. Martin et al. developed a learning environment called IPRO which works on the iPod Touch with a visual programming language [13]. Wolfgang proposed a visual programming system which works on Android smartphones for children [14]. Tillmann et al. proposed a programming environment which works on smartphones with a simple programming languages based on English [15]. Their system has a smart editor which suggests next inputs similarly to integrated development environments.

Although their environments work on smartphones well, they have no appealing user interface and image in their programming languages which motivates users to use and learn programming languages. However, our system have an appealing user interface with an icon-based programming languages including appealing icons, thus, it is more effective to motivate for learning programming.

VIII. Conclusion and Future Work

We developed a novel learning system using an appealing user interface with an icon-based programming languages. Our system motivates university students to learn programming and aids to learn three concepts of computational thinking. Although our current system uses an user interface and icons which a female university student designs for other female university students, our system can switch them to other user interfaces and other icons for suitable other people. Eight female university students improved their impressions of programming and their motivations for learning programming, and results of our achievement test by using our system in our experiment. Therefore, our system is effective and useful to learn programming and computational thinking.

In future, we will prepare other user interfaces and icons for other students and will confirm user interfaces and icons which students like are different and learning effectiveness depends on user interfaces and icons. Moreover, we will add new feature to convert our icon-based programming language to real programming languages such as Java, then, will help students to migrate from our programming language to real programming languages.

Acknowledgments

This research was partially supported by the Benesse Corporation and the Leave a Nest Co., Ltd.. We would like to thank Akari Nozawa, Aya Harashima and Daichi Katayama for their help with this research.

References

[1] R. Ballagas, F. Memon, R. Reiners, and J. Borchers, "istuff mobile: rapidly prototyping new mobile phone interfaces for ubiquitous computing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '07. ACM, 2007, pp. 1107–1116.

[2] J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, pp. 33–35, Mar. 2006.

[3] M. Felleisen and S. Krishnamurthi, "Viewpoint: Why computer science doesn't matter," *Commun. ACM*, vol. 52, no. 7, pp. 37–40, Jul. 2009.

[4] K. Redmond, S. Evans, and M. Sahami, "A large-scale quantitative study of women in computer science at stanford university," in *Proceeding of the 44th ACM technical symposium on Computer science education*, ser. SIGCSE '13. ACM, 2013, pp. 439–444.

[5] L. Nielsen Co., "Report on pilot data of use trend of android smartphones (in japanese)." [Online]. Available: http://www.netratings.co.jp/news_release/2012/12/Smartphone20121210.html

[6] A.-D. INC., "The device most frequently used by teenagers for accessing the internet is a smartphone (in japanese)." [Online]. Available: http://marketing.itmedia.co.jp/mm/articles/1301/30/news129.html

[7] I. R&D, "Report on rate of utilization of smartphones (in japanese)." [Online]. Available: http://www.impressrd.jp/news/121120/kwp2013

[8] H. Igo, Y. Hara, S. Matue, and C. Yoshimoto, "Introduction of nadeshiko programming language to education (in japanese)." [Online]. Available: http://www.edu.yamaguchi-u.ac.jp/~mis/www-page/mis/kaisetu/sotsuron2007/n-ihmy-main.pdf

[9] V. Cho, T. E. Cheng, and W. J. Lai, "The role of perceived user-interface design in continued usage intention of self-paced e-learning tools," *Comput. Educ.*, vol. 53, no. 2, pp. 216–227, Sep. 2009.

[10] H. Nittono, M. Fukushima, A. Yano, and H. Moriya, "The Power of Kawaii: Viewing Cute Images Promotes a Careful Behavior and Narrows Attentional Focus," *PLoS ONE*, vol. 7, no. 9, pp. e46 362+, Sep. 2012.

[11] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, Nov. 2009.

[12] S. Esper, S. R. Foster, and W. G. Griswold, "On the nature of fires and how to spark them when you're not there," in *Proceeding of the 44th ACM technical symposium on Computer science education*, ser. SIGCSE '13. ACM, 2013, pp. 305–310.

[13] T. Martin, M. Berland, T. Benton, and C. P. Smith, "Learning programming with ipro: The effects of a mobile, social programming environment," *Journal of Interactive Learning Research*, vol. 24, no. 3, pp. 301–328, July 2013.

[14] W. Slany, "Catroid: a mobile visual programming system for children," in *Proceedings of the 11th International Conference on Interaction Design and Children*, ser. IDC '12. New York, NY, USA: ACM, 2012, pp. 300–303.

[15] N. Tillmann, M. Moskal, J. de Halleux, and M. Fahndrich, "Touchdevelop: programming cloud-connected mobile devices via touchscreen," in *Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software*, ser. ONWARD '11. New York, NY, USA: ACM, 2011, pp. 49–60.