Wong, L.-H. et al. (Eds.) (2013). Proceedings of the 21st International Conference on Computers in Education. Indonesia: Asia-Pacific Society for Computers in Education

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

Kazunori SAKAMOTO[a*], Koichi TAKANO[b], Hironori WASHIZAKI[c] & Yoshiaki FUKAZAWA[c]
[a]*National Institute of Informatics, Japan*
[b]*OBIC Co., Ltd., Japan*
[c]*Waseda University, Japan*
*exkazuu@nii.ac.jp

Abstract: Computational thinking is one of the most important skills for using computers. Most existing learning systems for computational thinking work only on desktop or laptop computers, although the popularity of smartphones has rapidly been growing. Moreover, most existing programming languages to teach are based on English and most learning systems employ poor user interfaces. Thus, such programming languages and learning systems are not suitable for users who are not familiar with English or who are enchanted to such user interfaces.

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 employs an appealing interface that a female student designs for other female students and icons to motivate university students to learn programming through playing. We conducted an experiment with 16 female students from Waseda University to evaluate our system. We confirmed our system motivated the students to learn programming and helped learn computational thinking concepts.

Keywords: Learning system, computational thinking, programming language, smartphone

1. Introduction

The achievement of ubiquitous computing will become reality due to the wide use of smartphones (Ballagas, et al., 2007). Although the spread of computers gives birth to the necessity of computational thinking (CT) (Wing, et al., 2006), investigation results of the ACM show the relative unpopularity of computer science compared to more traditional mathematical disciplines (Felleisen, et al., 2009).

To motivate students for learning programming, we propose a gamified learning system using an appealing user interface (UI) with a novel icon-based non-verbal programming language. We summarize five features of our learning system as follows.

- **Affinity for smartphones**: Our system is an Android application working on smartphones instead of desktop or laptop computers (PCs). Smartphones are more popular and familiar than PCs with this demographic.
- **Game**: Our system provides a game that requires users to write programs in our icon-based language. Users unconsciously learn CT concepts through playing our system.
- **Appealing UI**: Our system employs appealing graphics for our UI and our icon-based programming language. To conduct an experiment to Japanese university students, our system employs graphics that a Japanese female university student designs for other students. Such graphics motivate students to play a game and learn CT.

- **Icon-based non-verbal programming language**: Our language consists of only 11 icon images, which indicate the user avatar's actions. Users can more easily write programs using icons than text.
- **Verbal programming language based on Native language**: Our system converts our icon-based languages into our verbal programming languages mutually so that users can confirm what the icons mean. To conduct an experiment to Japanese university student, our system employs Japanese-based verbal programming language.

2. Related Work

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 UIs. In this section, we show several existing work related to the obstacles and discuss about our system design to remove the obstacles.

2.1 Learning System for Programming and Computational Thinking on PCs

There have been studies on how to encourage young girls to learn programming and CT (Resnick, et al., 2009), (Esper, et al., 2013). Scratch is a programming environment that enables young people to create their own interactive stories, games, and simulations (Resnick, et al., 2009). The core users are between the ages of eight and 16. Young people can learn seven CT concepts: sequences, loops, parallelism, events, conditionals, operators, and data. CodeSpells is a video game that enables young people to learn Java programming (Esper, et al., 2013). 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 them is confirmed in their experiments, their environments work on only PCs. Our system concentrates on smartphones to motivate more students because the popularity of smartphones has rapidly been growing in Japan and US (Smith, 2012), (Nielsen Co., Ltd., 2012).

2.2 Learning System for Programming and Computational Thinking on Smartphones

There also have been studies on learning systems with smartphones. White, et al. utilize smartphones as computing platforms in education (White, et al., 2011) Martin et al. developed a learning system called IPRO that works on the iPod Touch with a visual programming language (Martin, et al., 2013). Wolfgang proposed a visual programming system that works on Android smartphones for children (Slany, et al., 2012). Tillmann et al. proposed a programming system that works on smartphones with a simple programming languages based on English (Tillmann, et al., 2011). Their system has a smart editor that suggests next inputs similarly to integrated development environments (IDE).

Although their systems work on smartphones well, their systems have no appealing UI that motivates users to learn programming languages. Our system has an appealing UI with an icon-based programming languages, thus, it is more effective to motivate for learning programming.

2.3 Icon-Based Educational Programming Language

Igo et al. conducted questionnaire study on reasons bachelor Japanese students from Yamaguchi University dislike programming (Igo, et al., 2007). 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 disliked 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. Thus, our system avoids using English for Japanese students.

2.4 Appealing User Interface

A UI is one of the most important things that motivate users to use the software. Cho et al. reported that good UI designs could motivate learners to continue using their e-learning tools (Cho, et al., 2009). Nittono et al. found that viewing such cute (or Kawaii in Japanese) images promoted careful behavior and narrow attentional focus regardless of gender (Nttono, et al., 2012).

An appealing UI (e.g. using Kawaii images) can motivate students to learn programming and can help students concentrate on programming. However, existing learning systems have no such UI. Thus, our system employs such an appealing UI for students.

3. Overview of Our Learning System

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

Each stage shows an example-dancing avatar (chicken). Users write a set of commands that replicates the dance steps. The user avatar (baby chick) and the example avatar (chicken) dance the same when the user's dance steps correctly replicate the example's. 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. Although the avatar graphics were designed for Japanese female university students, we can replace these images with other images designed for others.

Our system was developed as an Android application; therefore, users can learn CT 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.

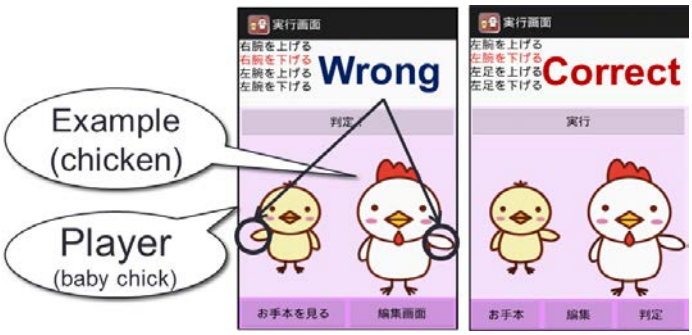

Figure 1. Screenshot of our Android application (wrong and correct answers on left and right sides).

4. Programming Language Design

We designed our programming language to address the following issues for students who are not familiar with English.

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

The left side of Figure 2 shows the Backus-Naur Form (BNF) of our language. The images in the BNF are icons similar to emoticons that are commonly used in mobile e-mail (Derks et al. 2007).

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 avatar's left/right wing or left/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.

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 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 nine action commands, the loop command, and ending sing of loops can be represented by icons or Japanese statements, which are interconvertible. That is, users can write programs with only icons and numbers for loops without character-based statements. The right side of Figure 2 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.

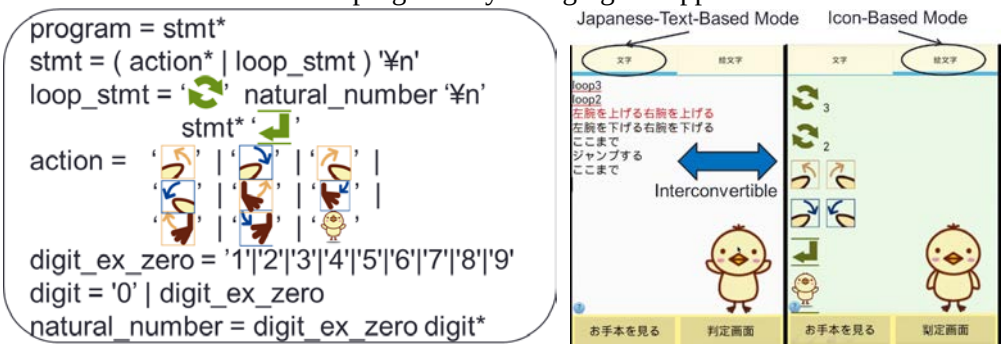

Figure 2. Backus-Naur Form of our icon-based programming language (on left side). Same program written using Japanese and numbers and using icons and numbers (on right side).

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 that 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 the right side of Figure 2 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.

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) Most 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.

4) Icons do not depend on characters so users can learn CT 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 just to explain what icons mean. Moreover, we can replace Japanese statements with other statements in other languages.

5. 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 Japanese female university students because the UI of our system is designed by a Japanese female university student for other Japanese female university students.

- RQ1: Does our learning system improve Japanese female university students' impressions of programming?
- RQ2: Does our learning system motivate Japanese female university students to learn programming?
- RQ3: Is our learning system more effective for learning three CT concepts than by reading lecture notes?

Sixteen Japanese female university students from Waseda University in Japan participated in our experiment. Nine out of the 16 were liberal arts students and seven were science students. We divided the 16 students into a group that used our learning system 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 system.

1. We gave a preliminary questionnaire to determine the students' impressions on programming and their motivations for learning programming.
2. The students used our learning system for approximately 40 minutes.
3. We conducted an achievement test for determining the depth of understanding of the three CT 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 system learned CT 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 system. 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).

The left side of Figure 3 is a box plot of the questionnaires results. The result indicates that the students' 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, no lowest points of the answers changed 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.

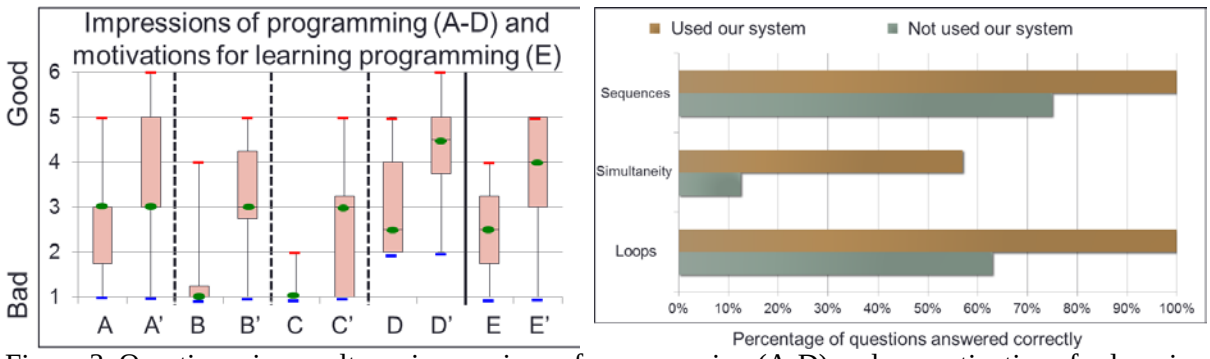

Figure 3. Questionnaire results on impressions of programming (A-D) and on motivations for learning programming (E), note that A-E and A'-E' indicate preliminary and post-investigation questionnaires (on left side). Percentages of questions answered correctly in achievement test (on right side).

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 CT concepts of sequences, simultaneity, and loops. We used a different game where the avatar's 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 that 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.

The right side of Figure 3 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 CT concepts more effectively compared with using lecture notes.

Therefore, we successfully improved Japanese female university students' impressions of programming (RQ1), motivated Japanese female university students to learn programming (RQ2). Our learning system was more effective for learning three CT concepts than by reading lecture notes (RQ3).

6. Conclusion and Future Work

We developed a novel learning system using an appealing UI with an icon-based programming language. Our system motivates university students to learn programming and aids to learn three concepts of CT. Although our current system uses a UI and icons which a female university student designs for other female university students, our system can switch them to other UIs 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 CT.

In future, we will prepare other UIs and icons for other students and will confirm that UIs and icons that students like are different depending on ages, genders or cultures. Moreover, we will confirm learning effectiveness depends on UIs and icons. We will also 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. Finally, as another direction for future work we will connect our learning system with toys such as robots that dance using our programming language so that our system try to catch more students' interest.

Acknowledgements

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.

References

Ballagas, R., Memon, F., Reiners, R., & Borchers, J. (2007). iStuff Mobile: Rapidly Prototyping New Mobile Phone Interfaces for Ubiquitous Computing. *Proceedings of CHI '07* (pp. 1107-1116). ACM Press.

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

Felleisen, M., & Krishnamurthi, S. (2009). Viewpoint: Why Computer Science Doesn't Matter. *Communications of the ACM*, *52*(7), 37-40.

Resnick, M., Maloney, J., Monroy-H. A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., and Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for All. *Communications of the ACM*, *52*(11), 60-67.

Esper, S., Foster, S. R., & Griswold, W. G. (2013). On the Nature of Fires and How to Spark Them When You're Not There. *Proceedings of CSE '13* (pp. 305-310). Denver, Colorado: ACM Press.

White, J., & Turner, H. (2011). Smartphone computing in the classroom. Pervasive Computing, IEEE, 10(2), 82-86.

Smith, A. (2012). 46% of American adults are smartphone owners: smartphone users now outnumber users of more basic mobile phones within the national adult population. Pew Research Center. http://pewinternet.org/~/media/Files/Reports/2012/Smartphone%20ownership%202012.pdf.

Nielsen Co., Ltd.. (2012). Report on Pilot Data of Use Trend of Android Smartphones (in Japanese). http://www.netratings.co.jp/news_release/2012/12/Smartphone20121210.html

Martin, T., Berland, M., Benton, T. & Smith, C.P. (2013). Learning Programming with IPRO: The Effects of a Mobile, Social Programming Environment. *Journal of Interactive Learning Research*, *24*(3), 301-328. Chesapeake, VA: AACE.

Slany, W. (2012) Catroid: A Mobile Visual Programming System for Children. *Proceedings of IDC '12* (pp. 300-303). Bremen, Germany: ACM Press.

Tillmann, N., Moskal, M., de H. J., & Fahndrich, M. (2011). TouchDevelop - Programming Cloud-Connected Mobile Devices via Touchscreen. *Proceedings of ONWARD '11* (pp. 49-60). Portland, Oregon: ACM Press.

Igo, H., Hara, Y., Matue, S. & Yoshimoto, C. (2007). Introduction of Nadeshiko Programming Language for Education (in Japanese). http://www.edu.yamaguchi-u.ac.jp/~mis/www-page/mis/kaisetu/sotsuron2007/n-ihmy-main.pdf

Cho, V., Cheng, T.C. E., & Lai, W.M. J. (2009). The Role of Perceived User-Interface Design in Continued Usage Intention of Self-Paced E-Learning Tools. *Computers & Education*, *53*(2), 216-227.

Nittono, H., Fukushima, M., Yano, A., & Moriya, H. (2012). The Power of Kawaii: Viewing Cute Images Promotes a Careful Behavior and Narrows Attentional Focus. *PLoS ONE*, *7*(9), e46362.

Derks, D., Bos, A. E.R., & Grumbkow, J. (2007). Emoticons and Social Interaction on the Internet: The Importance of Social Context. *Computers in Human Behavior*. *23*(1). 842-849.