Guided Debugging Practices of Game Based Programming for Novice Programmers

Chiung-Fang Chiu and Hsing-Yi Huang

***Abstract*—The objective of this study is to use game based programming to facilitate the teaching of debugging for novice programmers. The programing errors which novice programmers frequently committed were involved in the game programs. Worksheets were designed to guide students how to apply debugging strategies to find these errors and correct them. The debugging practices include the programming concepts of variable assignments, boolean statements, if statements and loop statements. Forty-one senior high school students participated in this experiment for six weeks including pre- and post-achievement test. The data including programming achievement test, debugging self-efficacy and questionnaire results were collected and examined. Though students' debugging self-efficacy wasn't significantly enhanced after the experiment, the paired-samples T test results show that this model of debugging practices was effective in improving students' programming concepts. Furthermore, students showed positive attitudes to this learning model and programming learning in the future.**

***Index Terms*—Computer science education, debugging practice, game-based programming, scratch.**

I. Introduction

The difficulties faced by many novice programmers have been documented in the literature. They usually don't know how to solve the programming programs [1], or feel disappointed when they see the incorrect program executing results [2]. Besides, how to debug is also an important issue. Novice programmers usually don't know how to find the program errors and correct them [3]. In the research report of [4], it mentioned that the time of debugging, testing and verification usually takes 50% to 70% of the program development process. Lahtinen, Ala-Mutka and Järvinen [5] also indicated that finding errors is the third difficult work when novice programmers learn programming, which is next to using programming to solve particular task, and recognizing the functionality of procedures. Though debugging is an important skill of learning programming, computer textbooks seldom provide related content [3]. Therefore, the design and development of learning content to improve debugging skill needs to be taken more attention to. If well-designed and interesting content can be provided to support teaching and learning of debugging, it can not only reduce the teaching load of the course instructors, but also promote the learning motivation of novice programmers.

Manuscript received March 15, 2014; revised May 23, 2014. This work was supported by the National Science Council of Taiwan, ROC, under the Grant numbers NSC 102-2511-S-260-002.

The authors are with the Graduate Institute of Curriculum Instruction and Technology, National Chi-Nan University, Taiwan (e-mail: cfchiu@ncnu.edu.tw, s101435509@ncnu.edu.tw).

Learning how to debug programs has many advantages [6]. First, debugging involves problem-solving procedures. The experiences of debugging can enhance learners' problem-solving ability which is also important to be applied in other domains. The continuous sequences of finding errors and correcting them provide learners the opportunities to reflect on their thinking process that is useful to develop higher order learning skills. However, unlike experienced programmers who can quickly find out errors, novice programmers usually don't plan the debugging strategies in advance and lack correct debugging skills. As a result, it is common that they often use trial-and-error approach to debug so that it will waste more efforts. Systematic scaffolding is needed to guide novice programmers on how to apply debugging strategies.

Many researchers investigated the differences of debugging behaviors among experienced and novice programmers. Gugrty and Olson [7] indicated that experienced programmers can propose more accurate assumptions of the reasons and locations of errors than novice programmers do. The debugging strategies which experienced programmers usually apply are as follows: 1) forward reasoning to grasp program's objective and current program status, 2) backward reasoning to search and find out the clues from program's outputs. Furthermore, making use of previous debugging experiences to solve current program errors is also important. Vessey [8] observed expert programmers have a broader error-searching ability and systemic thinking ability. In addition to the debugging strategies mentioned above, other debugging techniques such as using extra output statements in the program to produce outputs, or using the single-step mode of the compiler's debugging tool to trace execution of the program are also valuable to be introduced to novice programmers [9]. It is believed that novice programmers can gradually improve their debugging skills if adequate debugging practices with systematic scaffolding were conducted to them.

In addition to systematic scaffolding, the interest of the novice programmers to learn debugging should also be taken into consideration when designing debugging practices. Many studies tried to improve the learning motivation and effectiveness of programming teaching through building games. In the coding to develop games, students can engage in a game's virtual context and produce interesting programming results when finishing. Apart from gaining programming concepts, it also increases much more interest [10]-[12]. The results of Cliburn and Milller's study [13] revealed that students preferred game-based coding assignments to traditional programming assignments. Moreover, from the view point of constructivism pedagogy, the creation and experiences of game programming also provide opportunities for students to develop and enhance

their programming concepts more deeply. Seaborn, Ei-Nasr, Milam, and Yung developed a game construction-based curriculum for a high school computer science course [14]. Positive effect reveals that the curriculum was effective for understanding computer science and game design concepts. Similar responses to the high school curriculum that uses the creation of computer games to integrate computer science, art, and design instruction in a project-based learning model also resulted in increased computer programming knowledge and self-confidence for students [15]. Furthermore, Becker [16] found that writing known games such as Minesweeper and Asteroids can help students understand the concepts of object inheritance more thoroughly. This teaching approach provided a hook to capture students' imagination and energy. Moreover, Leutenegger and Edgington proposed a game-first approach to teach basic programming concepts via game development in Flash and ActionScript as fundamental to learn C++ later [17]. The result shows that this approach improved students' understanding of basic concepts. These previous works have evaluated the use and benefits of game programming for understanding and interest in computer science, programming skills and related concepts. Nevertheless, little has been done to reach out to teach debugging by using game programming. Thus, this study tried to design game based examples in Scratch to facilitate the teaching of debugging for novice programmers.

Scratch provides a visual programming environment targeted for creation of interactive stories, animations, games, art as well as music applications [18]-[20]. Although the original design is for school children aged from 8 to 16, its usage has been spread to any age level. Different from traditional text-input programming, Scratch provides drag-and-drop programming environment which eliminates syntax errors and encourages experimentation. It can reduce the cognitive load for novice programmers when programming concepts are first introduced to them [21], [22]. Consequently, students can engage in problem solving and algorithm design rather than focus on syntax issues. That Scratch having interactive visual interface and media-rich programming environment is suitable for novice programmers. Due to its ease of use and understanding, Scratch has been used as a lead-in course to other advanced programming course. For example, Scratch was introduced to students in an introductory computer science course at Harvard before students learned Java programming [23]. Scratch was also used to facilitate the teaching of computer science concepts or software engineering for high school students [24], [25]. The above research results suggest that Scratch might be chosen for novice programmers to study programming debugging. Thus, this proposed debugging practices guided by worksheets on Scratch game programs involving frequently committed errors were conducted in class to facilitate the programming learning.

II. Research Method

A. Learning Content

The errors involved in the buggy game programs to be practiced and the debugging strategies guided to students in each week were summarized in Table I. Students took debugging practices by the scaffoldings of worksheets which contained a sequence of steps to guide students on how to apply the debugging strategies to find the program errors in the buggy programs and how to correct them.

TABLE I: The Errors Involved in Buggy Programs in Each Week

Week no.	Errors	Debugging strategies
Week 1	If statements Loop statements	Check program's structures Show variable's value
Week 2	Variable assignments Loop statements	Show variable's value Extra output statements
Week 3	Variable assignments If statements	Show variable's value Predict program's output
Week 4	Boolean statements Nested if statements	Predict program's output Extra output statements

B. Procedure

This study was conducted at one senior high school's class. Forty-one students who have learned basic Scratch programming were involved in this study. These students had few programming experiences before participating in this experiment. The entire experiment lasted for six weeks. In the first week, students took the pre-experiment achievement test. Guided debugging practices were conducted for next four weeks. Finally, students took the post-experiment achievement test in the sixth week. The pre- and post-achievements test were written exams designed to measure the comprehension of program instructions and program structures. The maximum score of the achievement test was 100 points.

C. Programming Debugging Self-Efficacy Scale

Programming debugging self-efficacy scale adapted from [26] was administered to students at the first week of the experiment. Furthermore, the same scale was administered again at the end of the experiment. The self-reported responses of this instrument can range from "strongly disagree" 1) to "strongly agree" 5).

III. Results and Discussion

A. Achievement Test

Descriptions of means and standard deviations on pre- and post-test on achievement were depicted in Table II. Paired-samples *T* test was conducted to measure the differences between the pre- and post-test scores of achievement. The results depicted in Table III (t=2.88, p=.006) reveal that there was a significant improvement in assessment scores over time. Therefore, guided debugging practices improved students' programming concepts.

TABLE II: Description of Means and Standard Deviations on Pre- and Post-Achievement Test (N=41)

	Mean	S.D.
Pre-test	72.01	10.37
Post-test	80.59	16.15

TABLE III: Results of Paired Samples T-Test on Achievement Test (N=41)

Mean Difference	S.D.	t	D.f.	Sig. (2-tailed)
8.56	19.05	2.88	40	.006*

*p<.05

B. Students' Responses on Programming Debugging Self-Efficacy Scale

Descriptions of means and standard deviations on pre- and post-test on programming debugging self-efficacy were listed in Table IV. Paired-samples T-test was conducted to measure the differences between the pre- and post-test scores of programming debugging self-efficacy. The results depicted in Table V (t=0.92, p=0.37) suggest that it didn't show significantly difference between the pre and post-test scores in self-efficacy for programming debugging.

TABLE IV: DESCRIPTION OF MEANS AND STANDARD DEVIATIONS ON PRE- AND POST-TEST ON PROGRAMMING DEBUGGING SELF-EFFICACY (N=41)

	Mean	S.D.
Pre-test	3.79	0.81
Post-test	3.90	0.65

TABLE V: RESULTS OF PAIRED SAMPLE T-TEST ON PROGRAMMING DEBUGGING SELF-EFFICACY (N=41)

Mean Difference	S.D.	t	D.f.	Sig. (2-tailed)
0.11	0.79	0.92	40	0.37

C. Students' Responses on Questionnaire

At the end of the experiment, students were asked to fill out a questionnaire to give subjective feedback on the study. Table VI summarizes their responses of questionnaire. The questionnaires were meant to gain insight into the following issues:

1) About the guided debugging activities (Q1~Q3)

As regards the appropriateness of the buggy game examples, most students indicated that the difficult degree of debugging practices is appropriate (Q1, 79% agree and strongly agree). Furthermore, 86% of students thought debugging practices of game programming is very interesting (Q2). In the open-ended question of the questionnaire, one student wrote:

"Learning debugging strategies from Scratch game based programming is much more interesting and easy than what I had imagined at first."

In terms of the time allotted for the debugging practices, most students felt the time is just right (Q3: 74% agree and strongly agree). Nevertheless, some students hoped to have more time to debug. The possible explanation might be that students with lower prior programming knowledge might spend more time to find the errors and correct them. In the open-ended question of the questionnaire, one student indicated:

"Though the worksheets can guide me how to debug, I still spend much time on trying the debugging strategies. More time to practice is needed for me."

Thus, designing different difficult degrees of buggy programs for novices with different levels of prior knowledge is worth to take into consideration in the future study.

2) About the helpfulness of worksheets (Q4~Q7)

With respect to the helpfulness of worksheets, a majority of the students had positive responses. They agreed the worksheets helped them find the errors in programs (Q4), directed them to apply proper debugging strategies to solve program errors (Q5), and helped them learn the debugging skills in Scratch (Q6). Overall, students felt the worksheets were well designed (Q7). In the open-ended question, one student indicated:

"In fact, the buggy programs are difficult for me to debug at first. After the guidance of worksheets and the assistance of teacher, I gradually get familiar with the debugging strategies."

3) About the helpfulness of debugging practices for debugging and programming (Q8~Q12)

When asked about the helpfulness of debugging practices for debugging and programming, a majority of the students agreed that guided debugging practices improved their debugging skill (Q8) and enhanced their confidence to debugging (Q9). This teaching model also brought them more confidence to solve the program errors (Q10). Meanwhile, they also felt their debugging skills have been improved (Q11). After the debugging practices, they had the willingness to find the program errors and solve them on their own (Q12). Therefore, from the questionnaire results it can be summarized that though students' debugging self-efficacy wasn't significantly increased after the experiment, their confidence in debugging and solve program errors was improved. In the open-ended question of the questionnaire, one student mentioned:

"I have never learned the debugging strategies before. I learn a lot from these classes and like these game programs. It's beneficial and interesting."

4) About the attitudes toward programming learning (Q13~Q15)

The changes of students' attitudes toward programming learning were also examined in the questionnaire. Seventy-five percentage of students agreed the debugging practices enhance their programming ability (Q13). More than 70% of students indicated that they had more confidence to learn debugging skills in other programming language after these debugging practices (Q14, 71% agree and strongly agree). These positive experiences brought about by the guided debugging practices increased their interest to learn other programming language later (Q15). From these positive responses, it can be observed that students had positive attitudes toward programming learning after the experiment.

IV. CONCLUSION

This study proposed a teaching model for the learning of programming debugging at the high-school level. Debugging practices guided by worksheets on game programs involving frequently committed errors were conducted in class to facilitate the programming learning. The programming concepts included in the debugging practices are as followings: variable assignments, boolean statements, if statements and loop statements. Meanwhile, debugging strategies including predicting program's output, showing variable's value, check program's structures and extra output statements were introduced to novice programmers through debugging game based programs in Scratch.

In summary, this study demonstrated a feasible approach for the effective instruction of debugging skills to

high-school student. Though guided debugging practices did not have a significant effect on students' debugging self-efficacy, positive responses and much more confidence to debugging were revealed from the questionnaire results. Therefore, it seems that students' confidence to learn debugging did be promoted after the experiment, despite a lack of statistical significance. Furthermore, students showed positive attitudes to this learning model and programming learning in the future.

TABLE VI: STUDENTS' RESPONSES OF QUESTIONNAIRE AND RESULTS (*N*=41)

Question	(Strongly) agree	Neutral	(Strongly) disagree
1. The difficult degree of debugging practice is appropriate.	79%	20%	2%
2. Debugging practices of game programming is very interesting.	86%	12%	2%
3. The time allotted for the debugging practices is just right.	74%	17%	10%
4. Worksheets helped me find the errors in programs.	81%	17%	2%
5. Worksheets guided me apply proper debugging strategies to solve program errors.	81%	15%	4%
6. Worksheets helped me learn the debugging skills in Scratch.	88%	7%	5%
7. Overall, I felt the worksheets were well designed.	83%	15%	2%
8. Guided debugging practices improved my debugging skill.	73%	22%	5%
9. Guided debugging practices enhanced my confidence to debugging.	71%	22%	5%
10. This teaching model brought me more confidence to solve the program errors.	81%	17%	2%
11. After the debugging practices, I felt my debugging skills have been improved.	83%	15%	2%
12. After the debugging practices, I am willing to find the program errors and solve them by myself.	71%	24%	5%
13. The debugging practices improve my programming ability.	75%	15%	10%
14. After the debugging practices, I have more confidence to learn debugging skills in other programming language.	71%	24%	5%
15. These guided debugging practices increase my interest to learn other programming language later.	78%	15%	7%

REFERENCES

[1] M. West and S. Ross, "Retaining females in computer science: A new look at a persistent problem," *Journal of Computing Science in Colleges*, vol. 17, no. 5, pp. 1-7, 2002.

[2] J. A. White, "Teaching adult novices to program with Visual BASIC," *Journal of Computer Science Education*, vol. 11, no. 2, pp. 15-19, 1997.

[3] R. McCauley, S. Fitzgerald, G. Lewandowski, L. Murphy, B. Simon, L. Thomas, and C. Zander, "Debugging: a review of the literature from an educational perspective," *Computer Science Education*, vol. 18, pp. 67-92, 2008.

[4] B. Hailpern and P. Santhanam, "Software debugging, testing, and verification," *IBM Systems Journal*, vol. 41, no. 1, pp. 4-12, 2002.

[5] E. Lahtinen, H.-M. Järvinen *et al*., "A study of the difficulties of novice programmers," *ACM SIGCSE Bulletin*, vol. 37, no. 3, pp. 14-18, 2005.

[6] T. Lapidot and O. Hazzan, "Song debugging: merging content and pedagogy in computer science education," *ACM SIGCSE Bulletin*, vol. 37, no. 4, pp. 79-83, 2005.

[7] L. Gugerty and G. M. Olson, "Comprehension differences in debugging by skilled and novice programmers," in *Empirical Studies of Programmers*, E. Soloway and S. Iyengar, Ed., Norwood, NJ: Ablex, 1986, pp. 13-27.

[8] I. Vessey, "Expertise in debugging computer programs: A process analysis," *International Journal of Man-Machine Studies*, vol. 23, no. 5, pp. 459-494, 1985.

[9] A. C. Benander and B. A. Benander, "An analysis of debugging techniques," *Journal of Research on Computing in Education*, vol. 21, no. 4, pp. 447-455, 1989.

[10] K. Becker and J. R. Parker, "All I ever needed to know about programming, I learned from re-writing classic arcade games," presented at the International Conference on the Future of Game Design and Technology, Michigan State University, East Lansing, Michigan, October 13-15, 2005.

[11] J. Bayliss, and S. Strout, "Games as a 'flavor' of CS1," in *Proc. 2006 ACM Technical Symposium on Computer Science Education*, 2006, pp. 500-504.

[12] M. Panitz, K. Sung, and R. Rosenberg, "Programming in CS0: A scaffolded approach," *Journal of Computing Sciences in Colleges*, vol. 26, no. 1, pp. 126-132, 2010..

[13] D. C. Cliburn and S. M. Miller, "Games, stories, or something more traditional: The types of assignments college students prefer," in *Proc. of 2008 ACM Technical Symposium on Computer Science Education*, pp. 138-142, 2008.

[14] K. Seaborn *et al*., "Programming, PWNed: Using digital game development to enhance learners' competency and self-efficacy in a high school computing science course," in *Proc. 2012 ACM Technical Symposium on Computer Science Education*, pp. 93-98, 2012.

[15] J. Edgington, R. Fajardo *et al*., "Using game creating for teaching computer programming to high school students and teachers," in *Proc. of 2009 ACM 19th Annual Conference on Innovation and Technology in Computer Science Education*, 2009, pp. 104-108.

[16] K. Becker, "Teaching with games: The minesweeper and asteroids experience," *Journal of Circuits, Systems and Computers*, vol. 17, no. 2, pp. 23-33, 2001.

[17] S. Leutenegger and J. Edgington, "A games first approach to teaching introductory programming," *SIGCSE Bulletin*, vol. 39, no. 1, pp. 115-118, 2007.

[18] Scratch. [Online]. Available: http://scratch.mit.edu/

[19] J. Maloney, K. Peppler, Y. Kafai, M. Resnick, and N. Rusk, "Programming by choice: Urban youth learning programming with scratch," in *Proc. 2008 ACM Technical Symposium on Computer Science Education*, pp. 367-371, 2008.

[20] J. Maloney *et al*., "Scratch: A sneak preview," in *Proc. 2004 International Conference on Creating, Connecting, and Collaborating through Computing*, pp. 104-109, 2004,.

[21] B. Kaucic and T. Asic, "Improving introductory programming with Scratch?" in *Proc. 2011 MIPRO International Conference*, 2011, pp. 1095-1100.

[22] S. Garner, "Learning to Program from Scratch," in *Proc. 2009 IEEE International Conference on Advanced Learning Technologies*, 2009, pp. 451-452.

[23] M. Armoni *et al*., "Learning Computer Science Concepts with Scratch," in *Proc. 2010 International Workshop on Computing Education Research*, 2010, pp. 69-76.

[24] U. Wolz, H. Leitner, D. Malan, and J. Maloney, "Starting with scratch in CS 1," in *Proc. 2009 ACM Technical Symposium on Computer Science Education*, pp. 2-3, 2009.

[25] P. A. Sivilotti and S. A. Laugel, "Scratching the surface of advanced topics in software engineering: A workshop module for middle school students," in *Proc. 2008 ACM Technical Symposium on Computer Science Education*, pp. 291-295, 2008.

[26] D. R. Compeau and C. A. Higgins, "Computer self-efficacy: Development of a measuer and initial test," *MIS Quarterly*, vol. 19, no. 2, pp. 189-211, 1995.

Chiung-Fang Chiu received the B.S., M.S., and Ph.D. degrees in information and computer education from National Taiwan Normal University, Taiwan in 1994, 1997, and 2009, respectively. She is currently an assistant Professor in the Graduate Institute of Curriculum Instruction and Technology, National Chi Nan University, Nantou, Taiwan. Her research interests include computer science education, educational technologies, and teacher education.

Hsing-Yi Huang received the B.S. degrees in accounting from Feng Chia University, Taiwan in 1996. He is currently an undergraduate student in the Graduate Institute of Curriculum Instruction and Technology, National Chi Nan University, Nantou, Taiwan. His research interests include computer science education, teaching material design, and educational technologies.