

An Interactive Example for Game-Based Programming Environment

Po-Yao Chao, Yuan Ze University, Taiwan
Yun-Jen Hu, Yuan Ze University, Taiwan

The Asian Conference on Technology in the Classroom 2015
Official Conference Proceedings

Abstract

Many researchers suggest that programming is beneficial to career and the development of problem solving skills. However, traditional teaching methods and learning environments for programming accentuate the difficulties of programming since they merely emphasize the syntax or features of programming language and they offer few support in acquiring programming strategies. With advance of computer games and simulation environments, game-based problem solving activities have potential in improving the competence of programming strategies by providing interactive examples. Therefore, the purpose of this paper is to incorporate features of interactive examples and game-based learning to develop a game-based learning environment for programming strategies. Learners develop their programming strategies with the help of interactive examples that guide the learners with procedures of problem solving steps and ask the learners to complete partial solutions for problems. This completion task is believed to benefit the motivation and performance of developing programming strategies. The influences of the game-based learning on learner behavior, strategies, and performance are also explored in the paper.

Keyword: interactive example, game-based programming environment, programming strategies

iafor

The International Academic Forum
www.iafor.org

Introduction

With the popularity of computer and advanced information technology, leaning programming has been a trend. According to the U.S.A Bureau of Labor calculated, 2010 to 2020 the number of job openings of programmer is expected to growth 30% substantially, the other job only growth 14%(Lockard & Wolf, 2012). On the other hand, since September 2014 coding has been introduced to the school timetable for every child aged 5-16 years old, making the UK the first major G20 economy in the world to implement this on a national level. According to the research, learning programming can enhance students' ability of problem solving (Baroody & Coslick, 1993). Moreover, programming is a way of thinking by using abstraction and decomposition to solve problems (Liu, Cheng, & Huang, 2011). Having the ability to programming will enhance the self-competitive (Kiczales et al., 1997). Many researchers suggest that programming is beneficial to career and the development of problem solving skills (Atkinson, Derry, Renkl, & Wortham, 2000; Renkl, 2005; Sweller & Cooper, 1985; Zhu & Simon, 1987).

Cognitive Load Theory (Sweller, 1988, 2006, 2010) suggests that learning should consider the learner's cognitive load instead of traditional teaching method which merely focus on practice. To novice, learning from examples is more effective than directly answer the question (Renkl, 2005). However, compared with nature science, which can clearly observed behavior of learning target, programming involves abstract concepts and dynamic execution processes. It is difficult to observe the sequence and status during the execution of program in the traditional learning programming environments. Making programming a visualized process will assist the description of the programming process and status (Kaila, Rajala, Laakso, & Salakoski, 2010). Research found that simulation-based learning environment can improve the ability of problem solving (Liu et al., 2011), and game-based learning can improve engagement and motivation (Perrotta, Featherstone, Aston, & Houghton, 2013). Therefore, to help learner understand abstract concepts and dynamic execution process. We develop an interactive example scaffolding based on our previous game-based programming environment, and explore the influence of this interactive example scaffolding for learners.

Game-Based Programing Environment

The game-based programing environment proposed in this paper is a simulation environment which learner's goal is to instruct a robot so that he can collect items on a farm effectively. Learners generate an instruction card containing a set of instructions to control the robot. Through solving task in the environment, learners gradually learn the programming concepts and skills. However, in this paper, we designed an interactive example mechanism which a programming problem and the corresponding sets of instructions as solution made by experts were employed to help novice programmers learn how to solve the problem by exploring the solution. The novices may examine the instructions and then observe the result of executing these instructions. They also may insert or delete instructions to test their hypotheses. To improve transfer we design tasks similar with the interactive examples to examine novices' problem solving after the novices finish the exploration of interactive examples. Learning with the interactive example mechanisms contains the following parts:

(1) Expert model

Each programming problem involved in the interactive example mechanism corresponds to a set of instructions made by programming experts. The instructions serve as expert model that train the novice programmers in terms of how to apply and implement specific programming strategies. Novice learners can carefully consider the planning of path the robot should move and review the segments of instructions generated by the experts. Through the programming executing, the novice learners observe the ways which an expert solve programming problem and consequently develop a preliminary understanding of the model.

(2) Formation of mental model

Novice learners develop their mental model by generate and test their hypotheses on the predication of the robot's actions. The learners may insert or modify instructions embedded in the interactive examples and then verify the effect of the modified instructions on the robot's reaction. Several functions enable the learners to generate and test their hypotheses. They edit the expert instructions to implement the hypotheses, which includes: modify instructions, insert instructions, copy instructions, delete instructions, and modify the instructions.

(3) Control for simulation

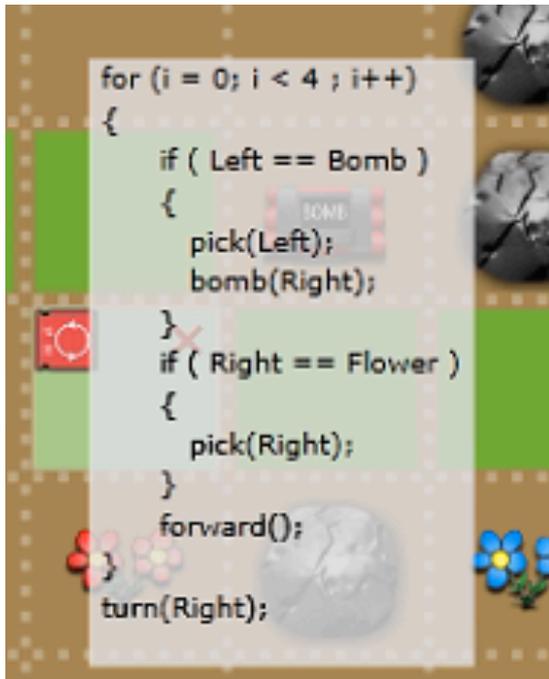
The simulation provides novice programmers with a continuous visual feedback. To help the learners build complete mental model, the mechanism also offers the functions to control the simulation. For example, when learners require further verification, they may set breakpoints to pause the execution of the program.

Illustration of Interactive Example

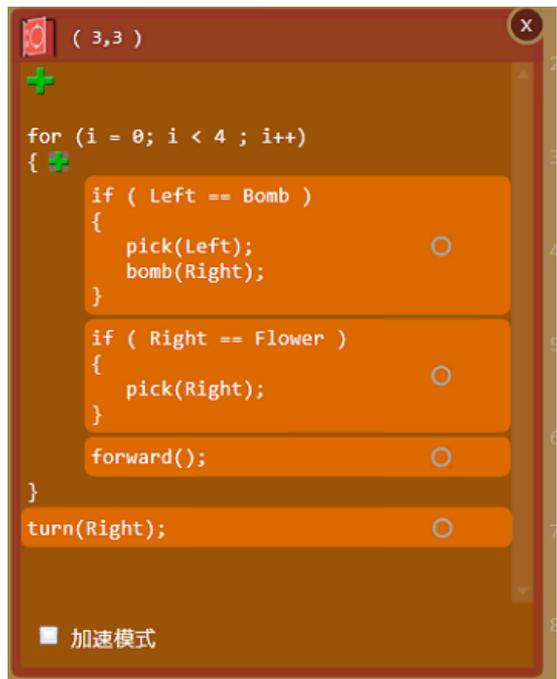
Figure 1 illustrates the use of interactive example. In this example a learner's goal is to collect all the flowers, bomb a stone and reach farmer's place. The initial state of the environment includes flowers, stones, and the places of robot and the farmer, as shown in Figure1 (a). The green path highlights the routes planned by the expert and instruction cards generate by the expert is displayed on the top left of a grid. In this example, expert set three instruction cards as the representation of corresponding solution. If a learner moves their mouse over the instruction cards, the content of that card will show next to it shown as Figure 1(b). Click the instruction card it will show the edit panel in the left side of scene shown as Figure1 (c).



(a)



(b)



(c)

Figure 1: Default environment

In this example, a learner first presses the execute button which initiates the robot to execute the instructions. The goal is to collect all of the flower and bomb a stone. An instruction card, as shown in Figure 2(a), includes a loop that repeats four times of a series of instructions. For each loop, the robot first determines whether there is a bomb on the left and then continues to determine whether there is a flower on its right. Finally the robot takes one step forward. The learner may hypothesize that the sequence of determination and the movement may make no difference in their corresponding results. As shown as Figure 2(a), the learner may try to modify the

sequence of the instruction by clicking the arrow next to the forward action to change its location as shown in Figure 2(b).

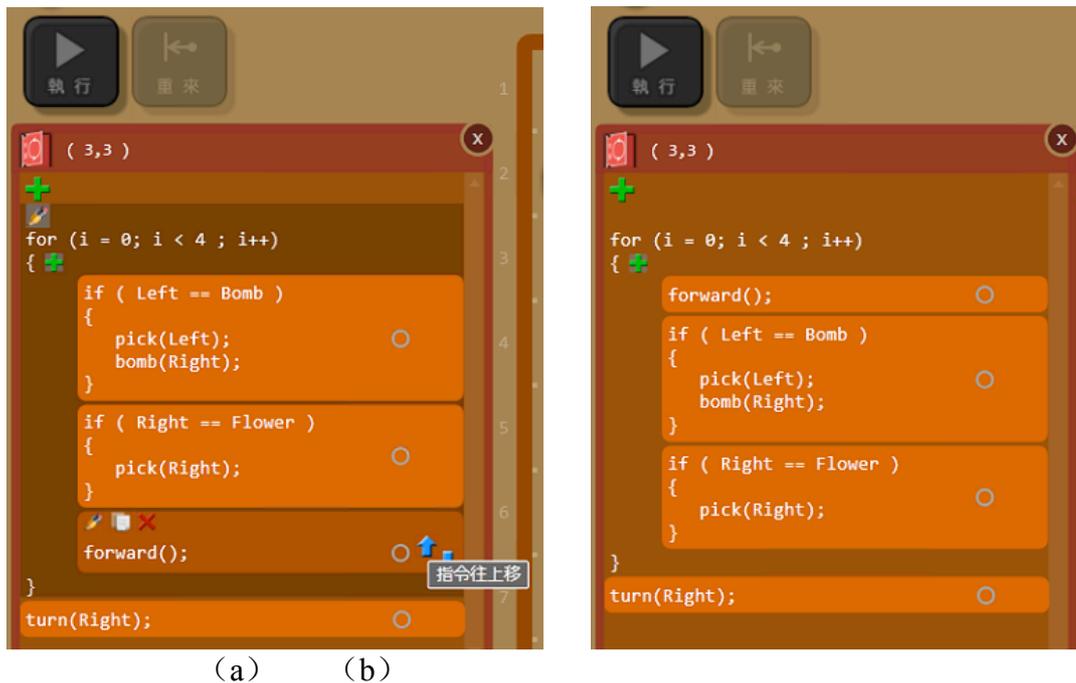


Figure 2: Move instruction

After executing the instructions contained in the Figure 2(b), the robot didn't pick a flower in coordinate (3, 4) shown in Figure 3(a). When robot reach the farmer's position, the system displayed messages indicating the failure of the program as shown in Figure 3(b). To fix the failure, the learner may observe the robot's behavior so that he can identify the problem. He set a breakpoint to purposeful pause the execution and examined the results.



Figure 3: Executed result

Evaluation

The participants are 84 freshman of non-major students who are 19 years old. Before evaluation, they are introduced to the interactive example mechanism for two hours. The process of the evaluation, shown in Figure 4, includes practice and survey phases. During the former phase, participants were asked to explore one interactive example and solve a programming problem similar with the provided example. This phase took 60 minutes. The latter phase aimed to explore how the participants use the interactive example mechanism by briefly interviewing the participants.

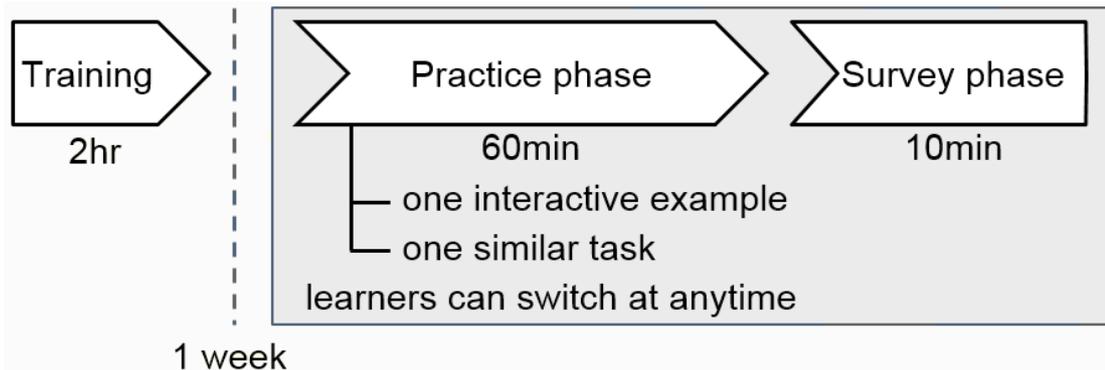


Figure 4: Process of evaluation

The preliminary result shows that 51 participants could finish the process of exploring interactive examples and solving the similar task successfully. This may suggest the interactive examples had the potential in assisting the participants solving specific programming problems by exploring corresponding interactive examples. The results of interview showed that some students think that learning programming through this game-based environment is more interesting than the conventional approach. For example, S11 said “learning programming from the game is a good way”. S25 said “It is more attractive and interesting than taking class”. With regard to the interactive example, some students think it provide guidance when they tackle with the similar tasks. For example, S13 noted “I knows how to solve the task when there is an example” and S34 considered more example can help learners reach task more successful. Some students give us useful advice, they suggest that we can give some hints to help them when they has difficulties with task. For example, S5 hope there is a hint when player spend too long to reach task. The student S6 said “I will not feel frustrated if there were a hint”.

Based on the result of the evaluation, the game-based learning environment and the proposed interactive example mechanism may help novice programmers solving programming problems and develop more complete mental model. We hope the proposed environment and mechanism can integrated into the curriculum so that novice learners can learning their programming in a more active and interesting manners.

Reference

- Atkinson, R. K., Derry, S. J., Renkl, A., & Wortham, D. (2000). Learning from examples: Instructional principles from the worked examples research. *Review of educational research, 70*(2), 181-214.
- Baroody, A. J., & Coslick, R. T. (1993). *Problem solving, reasoning, and communicating, K-8: Helping children think mathematically*: Prentice Hall.
- Kaila, E., Rajala, T., Laakso, M.-J., & Salakoski, T. (2010). *Effects of course-long use of a program visualization tool*. Paper presented at the Proceedings of the Twelfth Australasian Conference on Computing Education-Volume 103.
- Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M., & Irwin, J. (1997). Aspect-oriented programming *ECOOP'97—Object-oriented programming* (pp. 220-242): Springer.
- Liu, C.-C., Cheng, Y.-B., & Huang, C.-W. (2011). The effect of simulation games on the learning of computational problem solving. *Computers & Education, 57*(3), 1907-1918.
- Lockard, C. B., & Wolf, M. (2012). Occupational employment projections to 2020. *Monthly Lab. Rev., 135*, 84.
- Perrotta, C., Featherstone, G., Aston, H., & Houghton, E. (2013). *Game-based learning: latest evidence and future directions*: NFER Slough.
- Renkl, A. (2005). The worked-out-example principle in multimedia learning. *The Cambridge handbook of multimedia learning, 229-245*.
- Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive science, 12*(2), 257-285.
- Sweller, J. (2006). The worked example effect and human cognition. *Learning and instruction, 16*(2), 165-169.
- Sweller, J. (2010). Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educational psychology review, 22*(2), 123-138.
- Sweller, J., & Cooper, G. A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction, 2*(1), 59-89.
- Zhu, X., & Simon, H. A. (1987). Learning mathematics from examples and by doing. *Cognition and Instruction, 4*(3), 137-166.

Contact email: poyaochao@saturn.yzu.edu.tw

Acknowledge

This research was partially funded by the Ministry of Science and Technology under MOST103-2511-S-155-001-MY2