A Video Game-Like Approach to Supporting Novices in Learning Programming

Ami Sakakibara[1] and Hiroshi Hosobe[1]

Faculty of Computer and Information Sciences,
Hosei University, Tokyo, Japan
hosobe@acm.org

Abstract. In the education of introductory programming, people often adopt block-based visual programming languages such as Scratch and Blockly that allow programmers to construct programs by placing visual blocks. A previous study showed that a block-based language was more effective than a text-based language in introductory programming education. However, even with such block-based languages, it is still necessary for novices to learn programming in traditional ways, for example, by hearing lectures, reading textbooks, or watching tutorial videos. In this paper, we propose a video game-like approach to supporting novices in learning programming. We introduce two concepts into a block-based programming system: one is a staging mechanism that allows novices to gradually obtain more complex means of programming; the other is an assistant chatbot that helps novices to gain knowledge of programming. We implemented the system by applying our approach to turtle graphics. We present results of the experiment that we conducted to evaluate our approach.

Keywords: Block-based visual programming · Programming learning · Gamification.

1 Introduction

Programming education is being actively conducted throughout the world to increase students interested in computer science and to acquire excellent human resources for the information technology industry. In the education of introductory programming, people often adopt block-based visual programming languages such as Scratch [7, 10] and Blockly [9] that allow programmers to construct programs by placing visual blocks. A previous study showed that a block-based language was more effective than a text-based language in introductory programming education [12]. However, even with such block-based languages, it is still necessary for novices to learn programming in traditional ways, for example, by hearing lectures, reading textbooks, or watching tutorial videos.

In this paper, we propose a video game-like approach to supporting novices in learning programming. We introduce two concepts into a block-based programming system: one is a staging mechanism that allows novices to gradually

obtain more complex means of programming; the other is an assistant chatbot that helps novices to gain knowledge of programming. We implemented the system by applying our approach to turtle graphics [1]. We present results of the experiment that we conducted to evaluate our approach.

The rest of this paper is organized as follows. Section 2 describes previous work related to our approach. Section 3 proposes our approach, and Section 4 gives its implementation. Section 5 presents results of the experiment, and Section 6 discusses the approach. Finally, Section 7 provides conclusions and future work.

2 Related Work

One of the most related work was done by Arawjo et al. [2]. They proposed a progression design of a visual programming system that could be seen as being similar to our staging mechanism. However, they adopted functional programming, which is very different from block-based programming that is often used for introductory programming education.

Game-like approaches have been adopted also in the visual programming community. Bauer et al. [3] developed a block-based programming game called Dragon Architect to directly teach computational thinking strategies. Malizia et al. [6] developed a game-based system called TAPASPlay to foster computational thinking skills, focusing on playfulness and collaboration. Taylor et al. [11] developed a toolkit called IntelliBlox to enable learners to create block-based programs in immersive game-based learning environments. Lytle et al. [5] developed a game called Resource Rush to allow users to learn programming in open-ended game environments.

Fujimoto et al. [4] discussed research trends in game-based learning and open education. Open education refers to practice that eliminates barriers from education and increases educational opportunities. They recognized these two areas as becoming increasingly popular in the next few years.

Mineuchi et al. [8] developed a chatbot-based tool to allow students to easily perform preparation and review of their lessons by using a communication tool called LINE. They claimed that it would enable the students to increase opportunities for learning, to prepare without fear of failure, and to organize their thoughts.

3 Proposed Approach

We propose an approach to supporting novices in learning programming. The characteristic of our approach is that it allows novice users to enjoy learning programming as if they play introductory parts of video games. For this purpose, we particularly introduce the following two concepts into a block-based visual programming system.

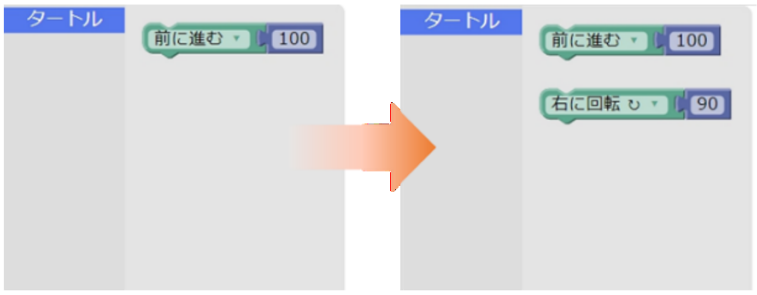

Fig. 1. Staging mechanism that gradually increases the available types of blocks as the user achieves goals for stages.

Staging mechanism: The system initially limits the types of visual blocks that the user can employ to construct a program, and it gradually increases the available types of blocks as the user achieves goals for stages (Figure 1).

Assistant chatbot: The system provides a chatbot that assists the user in learning programming by him/herself. As a help function, it explains in detail how to use blocks, and also it communicates with the user by making simple conversations.

Figure 2 shows the initial screen of our visual programming system. It allows the user to construct a block-based visual program for turtle graphics on part (a) of the screen, and presents the resulting graphics on part (b). On part (c) of the screen, it shows the points that correspond to the current stage. On part (d), it presents available blocks for the current stage. The user can call a chatbot by pressing button (e), which opens window (f) and makes the chatbot talk to the user.

3.1 Staging Mechanism

The staging mechanism increases the available types of blocks when the user achieves a given goal. It currently presents the following four stages:

Stage 1: Draw a straight line.
Stage 2: Draw a polygon.
Stage 3: Draw a geometric shape with a "repeat" block.
Stage 4: Draw free shapes.

Each stage is associated with a goal. For example, the goal for stage 1 is "Draw a straight line." The current stage is indicated by the points shown on the screen (at (c) in Figure 2). When the user achieves a goal, 100 points are added. The user can ask the chatbot about goals and points.

At stage 1, only one block "move forward or backward" is available. When the user achieves the goal by using and executing ore or more blocks, points and

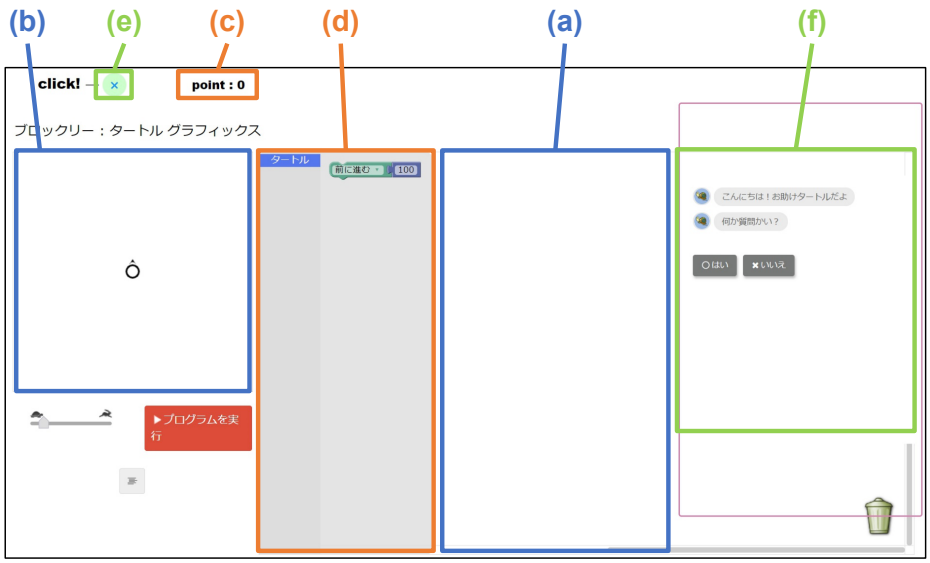

Fig. 2. Initial screen of our visual programming system.

a new block are added. When the system reads blocks to execute the program, it judges whether the goal has been achieved. If the goal has been achieved, it adds 100 points and reloads a page for stage 2 with an additional block.

At stage 2, two blocks including a new block "turn right or left" are available. At stage 3, a block "repeat" further becomes available. At stage 4, many types of blocks become available and can be freely combined.

3.2 Assistant Chatbot

Basically, the user communicates with the chatbot by selecting a message that is sent to the chatbot.

Help function. The user can ask the chatbot about various things such as how to use a block and how to effectively apply a block, by selecting "Help" from the menu that pops up when the user right-clicks a block. For example, when a block "move forward" is selected, a figure is presented to show that it draws a straight line, other related blocks also are given, and examples that can be drawn by combining other blocks are shown. Instead of immediately teaching the answer, it aims at enhancing the user's ability to think about how to achieve the goal.

Conversation function. Unlike the help function that can be asked about specific blocks, the conversation function answers more general questions such

as a question about the user interface. The conversation function of the chatbot is triggered when the button (e) in Figure 2 is pressed. It particularly aims at resolving questions of novice users who do not know what to do initially.

The conversation function always begins by asking whether there is any question. If there is no question, it finishes the conversation. Otherwise, it presents several choices to the user to identify what is the actual question. The first questions are about four things, i.e., "turtle graphics," "how to use," "points," and "blocks." After one of them is selected, it repeatedly presents more detailed questions. For example, if the user wants to ask about a block and selects the question about blocks, the user is prompted to use the help function. As another example, if the user selects the question about "how to use," it shows a screenshot of the user interface with numbers attached to its parts, which allows the user to select a part about which the user wants to ask. After reading the answer about the selected part, the user can return to the previous question to ask about another part.

4 Implementation

We implemented the system as a Web application by using Blockly [9] and its turtle graphics application. We used a JavaScript framework called BotUI to implement the assistant chatbot. The chatbot can be called by using a right-click pop-up menu as well as by pressing the button (e) in Figure 2.

The staging mechanism was implemented as reloading the page, which adds points and new blocks. Right after the reloading, the system displays a dialog box to inform that points have been added and new blocks have been introduced. When beginning stage 4, it additionally opens a pop-up window that informs that the final stage has been reached.

5 Experiment

We conducted an experiment to evaluate our approach. The purpose of the experiment was to examine whether the system based on our approach could improve its users' motivations for learning programming as well as whether it could reduce differences among the skills of the users caused by their past experiences in programming.

5.1 Participants

We recruited five participants (all male and 14.3 year old on average), three of whom had no experience in programming. In the following, we refer to them as participants A to E. Participants B and D had previous experiences in programming.

5.2 Procedure

We asked the participants to use our system. We measured the times that they spent to complete stages, and conducted questionnaires before and after the experiment.

Time measurement. To examine the influence of the past experiences of the participants in programming, we measured the times that they spent to complete each of the first three stages. For this purpose, we took them on video during the experiment. A camera was placed diagonally to the rear of each participant to make his hands and the computer screen visible to the camera.

Questionnaires. To examine the changes of the motivations of the participants for learning programming, we conducted questionnaires with the five-level Likert scale before and after the experiment. Both pre- and post-questionnaires included the following questions:

- Do you think that you are good at programming?
- Do you think that you enjoy programming if you learn programming in the future?
- Do you think that programming is difficult if you learn programming in the future?
- Do you positively want to learn programming in the future?

The post-questionnaire additionally included the following questions about our system:

- Did you use the chatbot?
- Was the chatbot easy to understand?
- Did you have a sense of achievement when you reached a goal?

The post-questionnaire also collected free descriptions of what difficulties the participants faced while using the system.

5.3 Results

We report the results of the experiment below.

Time measurement. Table 1 shows the times that the participants spent to complete stages 1, 2, and 3. Table 2 shows how many times they used the chatbots. The conversation function of the chatbot can answer the following questions: what is turtle graphics (indicated as "TG" in the table); I do not know how to use this user interface (indicated as "Interface" and with the numbers shown in Figure 3); I want to know about points (indicated as "Point"); I want to know about blocks (indicated as "Block"). Also, the help function of the chatbot can answer the following questions: how to move forward or backward (indicated

Table 1. Times that the participants spent to complete stages.

Participant	Stage 1	Stage 2	Stage 3	Total
A	0:01:19	0:16:27	0:02:34	0:20:20
B	0:01:31	0:06:16	0:04:22	0:12:09
C	0:05:39	0:34:18	0:06:59	0:46:56
D	0:13:20	0:00:52	0:04:00	0:18:12
E	0:02:57	0:22:56	0:40:57	1:06:50
Average	0:04:57	0:16:10	0:11:46	0:32:53

Table 2. Numbers of the uses of the chatbots by the participants.

Participant	Conversation										Help		
	TG	Interface							Point	Block	Move	Turn	Repeat
		1	2	3	4	5	6	7&8					
A			2	1				1		1			
B	1			1	1			1		3		1	
C	2		4	3	1	1	1	2					
D	1		2	1				2			1		
E		1	4	1			1	1			1		
Total	4	1	12	7	2	1	2	7	0	4	2	1	0

as “Move” in the table); how to turn right or left (indicated as “Turn”); how to repeat (indicated as “Repeat”).

The times of the participants A, B, C, and E show that stage 2 took longer than stage 1. By contrast, participant D spent a longer time at stage 1. This was probably because when the program should be executed was not explicitly explained. It was observed that participant D pressed the execute button before placing blocks in the workspace and did not press the execute button long after placing blocks. Also, participant D had a previous experience in programming, and did not spend particularly long times at stages 2 and 3.

Participants A, B, and C spent shorter times at stage 3 than at stage 2. This was probably because the goal for stage 3 was not very different from the goal for stage 2. By contrast, participants D and E spent longer times at stage 3 than at stage 2. This was probably because they initially thought that a repeat block could contain only one block. We prepared a help function about a repeat block that explains that it could contain one or more blocks. However, no participants used this help function as shown in Table 2.

Questionnaires. Table 3 shows the results of the questionnaires. The five-level Likert scale consists of the following: “strongly agree” as 5, “agree” as 4, “undecided” as 3, “disagree” as 2, and “strongly disagree” as 1. We performed a paired t-test between the scores of the pre- and the post-questionnaire. The re-

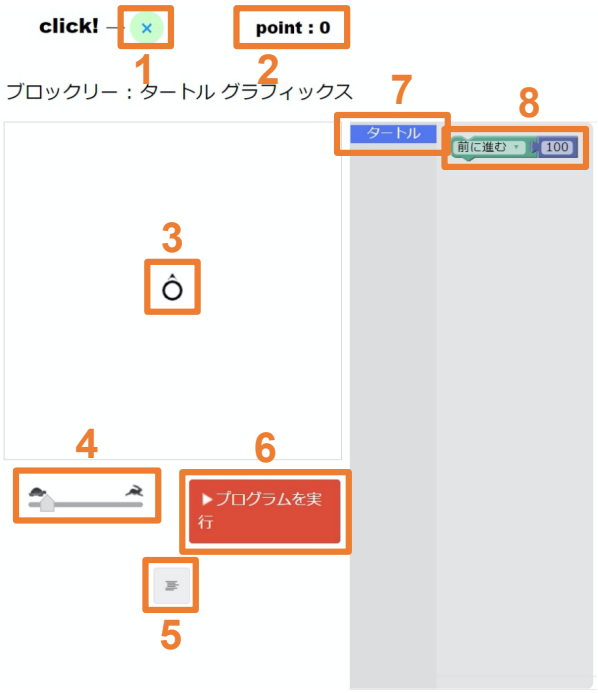

Fig. 3. User interfaces about which the chatbot can be asked.

sults show that there were no significant differences concerning the four questions described in Subsection 5.2. This means that our system did not bring sufficient psychological influences.

The questions about our system in the post-questionnaire showed the following. All the participants used the chatbot. Four participants strongly agreed to the question "Was the chatbot easy to understand?" and the other one agreed to it. Four participants agreed to the question "Did you have a sense of achievement when you reached a goal?"

In the post-questionnaire, participants A to E provided the following free descriptions of difficulties in using the system:

A. I did not know what I should do before the chatbot taught me about it.
B. I spent a little long time in repeating blocks.
C. I did not understand the goal "geometric shape" for stage 3.
D. It was hard to draw a polygon by combining various blocks in stage 3.
E. It was difficult to draw a polygon.

Table 3. Results of the questionnaires before and after the experiment.

	Good at programming		Enjoy programming		Programming is difficult		Want to learn programming	
Participant	Before	After	Before	After	Before	After	Before	After
A	3	4	4	5	4	4	5	5
B	3	4	4	5	3	2	3	5
C	2	2	4	3	3	5	3	2
D	4	4	4	5	4	4	3	3
E	2	1	5	4	4	5	5	4
Mean	2.8	3.0	4.2	4.4	3.6	4.0	3.8	3.8
Variance	0.7	2.0	0.2	0.8	0.3	1.5	1.2	1.7

6 Discussion

Although the experimental results did not statistically show psychological influences, differences among the participants' reactions and spent times were observed. A cause for the large differences among the novice participants was that the explanation about how to use a repeat block was insufficient. Although we had included the explanation in the help function, no participants used it during the experiment. It might seem plausible that the help function should be moved to the conversation function because the conversation function was relatively more used. However, this would largely increase choices when the number of block types would become large. Since there were participants who did not use the help function at all, the right-click pop-up menu is considered to be inappropriate. It might have been possible to trigger the help function by placing a block about which a user wants to know.

Our system enabled the participants to achieve the goals by themselves, which suggests that we were able to develop a programming system for novice programmers. However, our system is still not sufficient for improving computational thinking because it does not sufficiently support the participants in correctly using repeat blocks.

7 Conclusions and Future Work

We proposed a video game-like approach to supporting novices in learning programming. We introduced a staging mechanism and an assistant chatbot into a block-based programming system. The experiment that we conducted suggested the usefulness of the staging mechanism and the conversation function of the chatbot. However, it showed that the help function of the chatbot was not useful. This problem could be solved by simplifying the interface for triggering the help function. The experiment also suggested the necessity of a function for supporting users in understanding repeat blocks.

Acknowledgment

This work was partly supported by JSPS KAKENHI Grant Number JP17H01726.

References

1. H. Abelson, N. Goodman, and L. Rudolph. LOGO manual. AI Memo 313, AI Lab., MIT, 1974.
2. I. Arawjo, C.-Y. Wang, A. C. Myers, E. Andersen, and F. Guimbretière. Teaching programming with gamified semantics. In *Proc. ACM CHI*, pages 4911–4923, 2017.
3. A. Bauer, E. Butler, and Z. Popović. Dragon Architect: Open design problems for guided learning in a creative computational thinking sandbox game. In *Proc. Intl. Conf. on the Foundations of Digital Games (FDG)*, number 26, pages 1–6. ACM, 2017.
4. T. Fujimoto, K. Shigeta, and Y. Fukuyama. The research trends in game-based learning and open education. *Educ. Technol. Res.*, 39(1):15–23, 2016.
5. N. Lytle, J. Echavarria, J. Sosa, and T. W. Price. Resource Rush: Towards an open-ended programming game. In *Proc. Workshop on Blocks and Beyond*, pages 91–93. IEEE, 2019.
6. A. Malizia, T. Turchi, D. Bell, D. Fogli, and F. Danesi. Fostering computational thinking through collaborative game-based learning. *Multimed. Tools Appl.*, 78:13649–13673, 2019.
7. J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond. The Scratch programming language and environment. *ACM Trans. Comput. Educ.*, 10(4):16:1–15, 2010.
8. A. Mineuchi, R. Matsuba, M. Toda, and K. Suzuki. Design of an educational supporting tool for encouragement of learning with a chat bot. In *Proc. Annual Meeting of the Academic Exchange for Information Environment and Strategy*, number TF2-5, pages 1–6, 2017. In Japanese.
9. E. Pasternak, R. Fenichel, and A. N. Marshall. Tips for creating a block language with Blockly. In *Proc. Workshop on Blocks and Beyond*, pages 21–24. IEEE, 2017.
10. M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. S. Silver, B. Silverman, and Y. B. Kafai. Scratch: Programming for all. *Comm. ACM*, 52(11):60–67, 2009.
11. S. Taylor, W. Min, B. Mott, A. Emerson, A. Smith, E. Wiebe, and J. Lester. IntelliBlox: A toolkit for integrating block-based programming into game-based learning environments. In *Proc. Workshop on Blocks and Beyond*, pages 55–58. IEEE, 2019.
12. D. Weintrop and U. Wilensky. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Trans. Comput. Educ.*, 18(1):3:1–25, 2017.