A Scratch-Based Collaborative Learning System with a Shared Stage Screen

Yusuke Fukuma, Kumpei Tsutsui, Hideyuki Takada(✉), and Ian Piumarta

Faculty of Information Science and Engineering, Ritsumeikan University,
1-1-1 Noji-Higashi, Kusatsu, Shiga 525-8577, Japan
htakada@cs.ritsumei.ac.jp
http://www.cm.is.ritsumei.ac.jp/~htakada/index_e.html

Abstract. Opportunities for elementary and junior high school students to learn programming have been increasing over the last few years, and programming education in elementary schools will become compulsory in 2020. Applying 'collaborative learning' to programming education can be considered an effective method, but the current environment for collaborative learning in programming education needs to be improved because in many cases the creation and execution of programs are completed on isolated personal computers. To improve this situation we have developed a collaborative learning support system featuring a shared 'stage' screen based on the visual programming environment Scratch. Our system keeps each individual's Scratch programming environment separate from the shared stage screen, but allows each of the individual stages to be displayed together on the shared screen at the same time. The system was used during collaborative learning workshops at a local community center. Evaluations were made with post-workshop survey questionnaires and analysis of the learners' behavior. We confirmed that co-teaching and communication among learners occurred because the programs developed by others were visible on the shared stage screen.

Keywords: Collaborative learning · Programming education · Shared screen

1 Introduction

With the introduction of compulsory programming education in the Japanese standard curriculum for elementary schools in 2020 [3], increasing interest in programming education is gaining the attention of workshop organizers. Programming workshops for elementary and secondary school students are being held by various organizations, including non-profit organizations and companies.

At the same time, the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) is emphasizing collaborative learning as one of the key methods of education for the 21st century [2]. In collaborative learning, a teacher facilitates communication and mutual teaching between children during classes. Collaborative learning can deepen the students' understanding by

T. Yoshino et al. (Eds.): CollabTech 2017, LNCS 10397, pp. 84–98, 2017.
DOI: 10.1007/978-3-319-63088-5_8

encouraging them to exchange ideas, thoughts, and opinions with their peers. Through the advocacy of MEXT, various new methods of classroom teaching (such as collaborative learning) are being introduced to schools, in contrast to the 'one-way guidance' from teacher to students that was normal in the past.

Collaborative learning, involving teaching and learning within a group instead of traditional one-way guidance or individual learning, could also be an effective method for programming education. In most practical situations, however, students are isolated with their own computer on which they create and execute their programs. This is far from an effective environment for collaborative learning. Communication among students must also be promoted for effective collaborative learning, which is outside the scope of conventional methods.

To improve this situation we developed a collaborative learning system based on Scratch [5]. Scratch is a visual programming environment in which users create projects using multimedia and scripts. A typical Scratch user creates programs to control object movement on a 'stage' displayed on their personal computer, leading to isolated project work. We extended Scratch to be suitable for collaborative learning, by allowing multiple users' individually-created moving objects to be displayed on a single, shared stage. It is expected that seeing the movement of objects created by others on the shared stage will promote communication and co-learning between users. We evaluated the effectiveness of our system by using it during programming workshops with young students.

2 Collaborative Learning with Programming

In this section we describe some of the problems faced when introducing collaborative learning into programming education at elementary schools, in order to formulate the requirements for an effective system.

2.1 Programming Education in Elementary School

The MEXT suggests that programming education at elementary schools should address the following four points:

- Children notice that computers are being used in their daily lives.
- Children notice the necessity of solving problems by taking a series of steps.
- Children acquire *programming thinking*.
- Children try to make use of computers in their own lives.

To achieve these learning goals, the MEXT also suggests that it is important to create a method of programming education that encourages "subjective, interactive and deep learning", and that classes should never become silent with students just facing their own computers.

When considering a form of programming education based on the points above, we note that conventional one-way teaching cannot be subjective because in most cases it forces children to learn passively. Similarly, private one-on-one lessons cannot promote interaction among learners. The conventional method is therefore considered inappropriate.

In elementary school programming education, conventional classroom lessons or personal lessons should therefore be replaced by learning methods that promote inter-student interactions in addition to self-directed learning.

2.2 Problems in Collaborative Learning of Programming

To engage in collaborative learning, the following two conditions should be satisfied:

- Learners can share their thoughts through communication with others.
- Learners can incorporate the opinions of others and improve their own performance based on them.

In programming education, however, learners have little chance to be involved with others as they focus on using just the computer in front of them. Communication is subsequently less motivated, and opportunities to incorporate others' opinions are reduced.

To solve this problem it is necessary for learners to be exposed not only to their own project but also to the projects developed by others. By learning in an environment where they can take an interest in other projects, they are naturally encouraged to talk about their projects, ask questions, make suggestions, and teach each other new skills.

2.3 Related Work

The system described in this paper is an extension of Single Display Groupware (SDG) [6]. SDG enables co-located users to collaborate using a single, shared display while simultaneously using multiple input devices. In order to adapt SDG to a visual programming environment such as Scratch, we placed a single stage on a shared display screen while keeping the programming pane on the individual users' personal computers.

Some research work emphasizes collaborative scripting by multiple users for visual programming environments [4,7,8], but they are based on a 'multiple computers, multiple displays' paradigm.

3 Development of the System

In this section we first describe the requirements, functions and implementation of a collaborative learning system based on Scratch. We then describe a scenario for its use in a practical situation.

3.1 Requirements

As mentioned above, Scratch is essentially a single-user application. A single window displays three panes containing command blocks, scripts, and the stage. To make collaborative learning possible, the system should be modified to satisfy the following requirements:

- The stage pane, showing the results of executing scripts, can be shared.
- Both individual development and group development can be performed.

Based on these requirements, the stage pane (where scripted actions are displayed) is separated from the other panes and displayed on a different screen shared among the co-located users. As shown in Fig. 1, in addition to the individual development terminals used by the learners, a 'shared stage screen' displays all the stage panes belonging to the users within the collaborating group. Furthermore, as shown in Fig. 2, the shared stage can be unified into a single project by removing the frames separating one individual stage from the others, allowing objects from different stages to interact with each other on a single, shared stage.

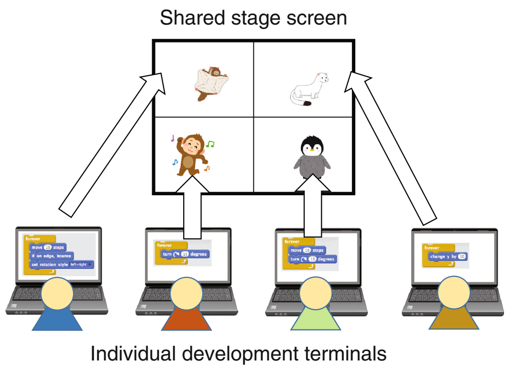


Fig. 1. Sharing the stages on the shared stage screen

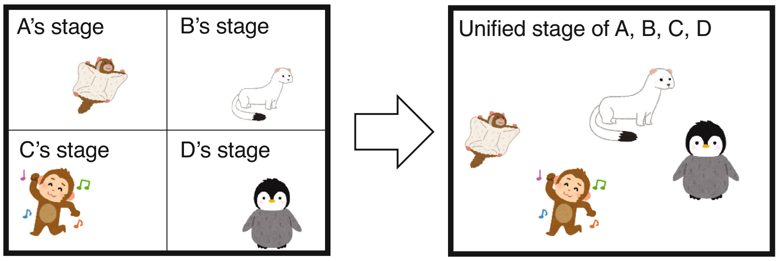


Fig. 2. Removing the frame on the shared stage screen

3.2 Functions

The system has four major functions: a ‘programming’ function allowing the user to create and execute programs, a ‘stage sharing’ function for displaying the multiple stages of projects developed on the users’ computers, a ‘group management’ function for choosing which individual computers belong to the shared stage group, and a ‘frame removal’ function to switch between multiple individual stages and a single, shared stage.

Programming. In Scratch, users can easily develop a program by combining prepared command blocks using drag-and-drop, and then see the resulting motion of their objects on the stage. Figure 3 shows the screen used for program development. It is an extended version of Scratch, displayed on the personal computer of an individual user.

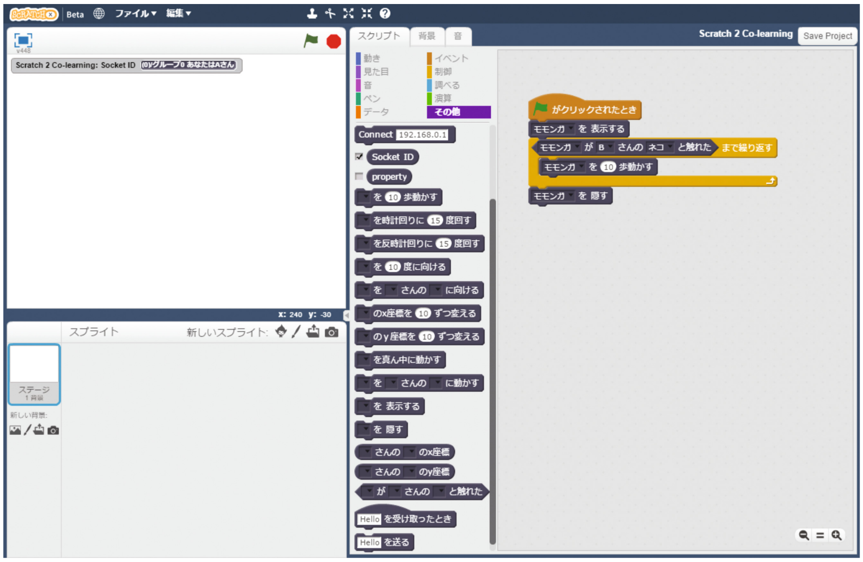


Fig. 3. Screen of the development terminal

In our system, to make scripted objects appear on the shared stage screen (rather than on the user’s own development terminal), special command blocks (shown in the center column on the screen) are used for programming. As shown in Fig. 4, the object to be controlled is chosen using a pull-down menu on the command block, and the action invoked by the command block will be sent to the server driving the shared stage instead of to the locally-displayed stage.

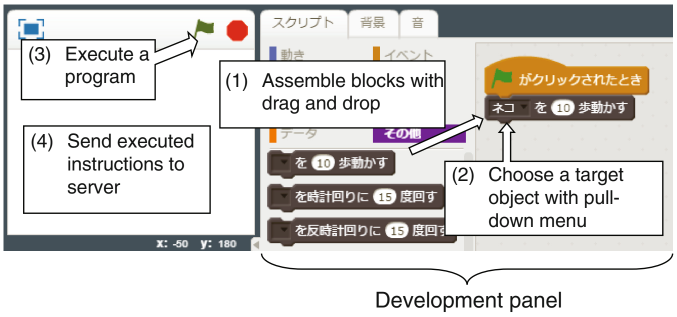


Fig. 4. Programming on our Scratch-based system

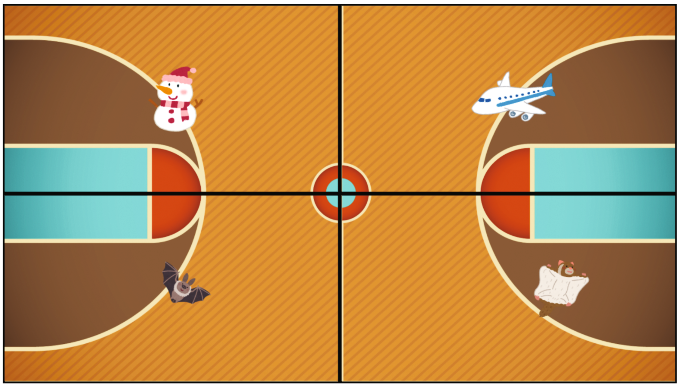
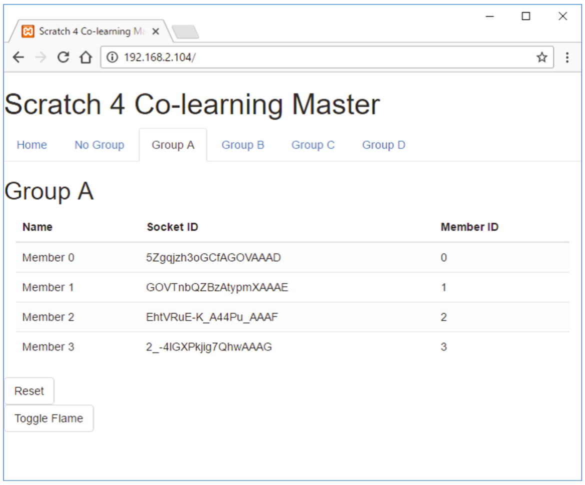


Fig. 5. Shared stage screen

Stage Sharing. Figure 5 shows the shared stage screen. A single display shows the stages from every user in the group. Frames divide the screen into four areas, each of which is assigned to display one of the user stages. In the example shown, each stage contains a single object which will move according to the actions generated from the script running on the user's personal computer.

Group Management. Figure 6 shows the group management Web interface, running on the server, allowing the system operator to manage group membership. The stages for individual development computers can be assigned to a specific quadrant of the group's shared stage screen.

Frame Removal. This server function is also provided by the Web interface, accessed with a Web browser. Pressing the 'toggle frame' button removes the frame separating the stages on the shared display. Removing the frame switches the system from individual development mode to group development mode. In


Scratch 4 Co-learning Master

Group A

Name	Socket ID	Member ID
Member 0	5Zgqjzh3oGCfAGOVAAAD	0
Member 1	GOVTnbQZBzAtypmXAAAE	1
Member 2	EhtVRuE-K_A44Pu_AAAF	2
Member 3	2_-4IGXPkjig7QhwAAAG	3

Fig. 6. Management console

group development mode, objects can move beyond the bounds of their 'home' stage and can be programmed to interact with objects created by other users.

3.3 Implementation

The structure of our system is shown in Fig. 7. Each group consists of four development terminals, one shared stage screen terminal, and a server. The development terminals and the shared stage screen terminal communicate with the server using the WebSocket API. Commands executed on the development terminals are transmitted to the shared stage screen terminal via the server. Object coordinates, and collisions between objects, are monitored on the shared stage screen terminal and transmitted back to the server. The server then forwards this information to the individual development terminals.

The individual development environments run in the users' Web browsers. The functionalities of the shared stage screen terminal and the server are provided by Windows applications. The server and the shared stage screen application can be operated on the same computer. Figure 8 shows a picture of the entire system in use.

Development Terminal. Special command blocks for interacting with objects on the shared stage screen were added using ScratchX [1], an extension of Scratch that facilitates the addition of new command blocks written in JavaScript. Our additional command blocks are shown in Table 1.

When a command block in the 'operation' category is executed, the development terminal does not perform the block's operation locally. Instead it transmits the attributes of the executed operation to the server, along with the ID

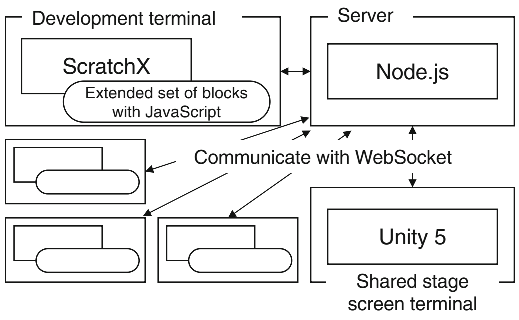

Development terminal
ScratchX
Extended set of blocks with JavaScript
Server
Node.js
Communicate with WebSocket
Unity 5
Shared stage screen terminal

Fig. 7. System structure

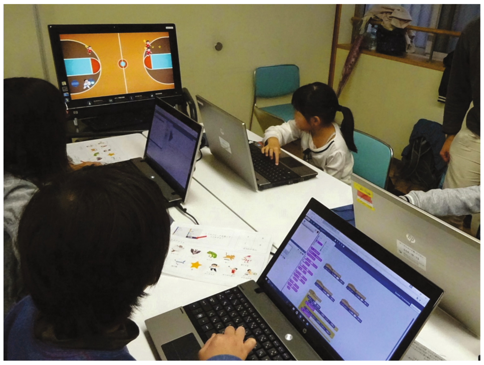


Fig. 8. Actual application environment

assigned to the development terminal. Command blocks in the 'monitoring' category are triggered in response to information received from the server about object coordinates and collisions.

Server. The server mediates communication between the development terminals and the shared stage screen terminal. It transmits executed commands from the development terminal to the shared stage screen terminal, and sends information about the monitored status of object coordinates and collisions received from the shared stage back to individual development terminals. The server is implemented using JavaScript, Node.js, and the Apache HTTP daemon.

Group membership is managed by the system operator using the group management interface. When a development terminal is added to a group, the server allocates a unique ID that will be associated with the terminal. Terminal IDs are stored on the server to allow group membership to be determined. When the server receives information from a development terminal, it uses the terminal's ID to identify its group. The information is then propagated to the other terminals belonging to the group.

Table 1. Extended set of blocks

Category	Block	Function
Connection	Connect [A]	Connect to server A
Operation	Move [A] [B] steps	Move object A in B steps
	Turn [A] clockwise [B] degrees	Turn object A clockwise in B degrees
	Turn [A] counter-clockwise [B] degrees	Turn object A counter-clockwise in B degrees
	Point [A] in direction [B]	Point object A in direction B degrees
	Point [A] towards [B]'s [C]	Point object A towards object C of user B
	Change [A]'s x by [B]	Add B to x-axis of object A
	Change [A]'s y by [B]	Add B to y-axis of object A
	Move [A] to center	Move object A to the center of the coordinate
	Move [A] to [B]'s [C]	Move object A to the coordinate of object C of user B
	Show [A]	Show object A in the stage
	Hide [A]	Hide object A in the state
	Send [A]	Send message A to all objects in the group
Monitoring	[A] touches [B]'s [C]	Check if object A touches object C of user B
	x position of [A]'s [B]	X-axis of object B of user A
	y position of [A]'s [B]	Y-axis of object B of user A
	When I receive [A]	An event receiving message A

Shared Stage Screen Terminal. The shared stage screen terminal is implemented using C# and the Unity engine. It renders users' objects according the information received from the server about 'operation' commands executed on the development terminals. It also monitors its own objects' coordinates and collisions, transmitting this information back to the server. Transmissions to the server are made every few tens of milliseconds. From the server, relevant parts of this information are forwarded back to the development terminals where it is used to control 'monitor' command execution.

3.4 Use Case

A typical scenario of multiple users using our system for collaborative programming is described below.

Every user executes the command 'Connect A' at their development terminal, causing the terminal to be connected to server A. After all terminals are connected, the operator forms a group by choosing four development terminals from the list of connected terminals. Each development terminal is assigned to a quadrant of the shared stage.

Figure 9 shows an example of individual development with our system. Users create and then execute a program. Their scripted objects are displayed on the shared stage screen, within the area assigned to their development terminal. Their objects cannot move beyond the frame separating their area from the three other areas.

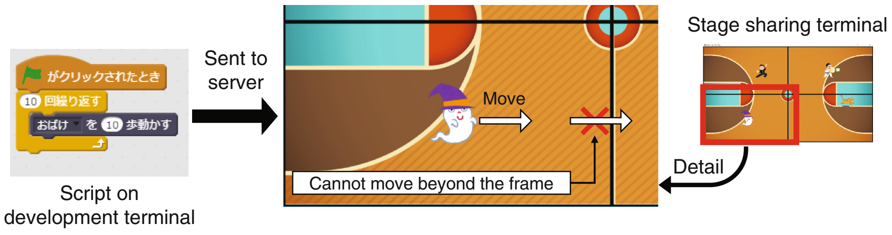


Fig. 9. Example of use for personal development

When the operator presses the ‘toggle frame’ button on the group management console, the frame separating the four user areas is removed and users can perform group development. In group development, objects are free to move over the entire shared stage as shown in Fig. 10. In addition, objects can be programmed to interact with other objects; for example, one user turns their ‘ghost’ object towards the ‘cat’ object created by another user, as shown in Fig. 11.

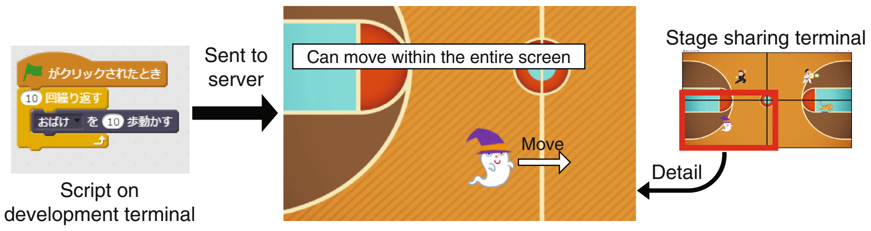


Fig. 10. Use case in group development

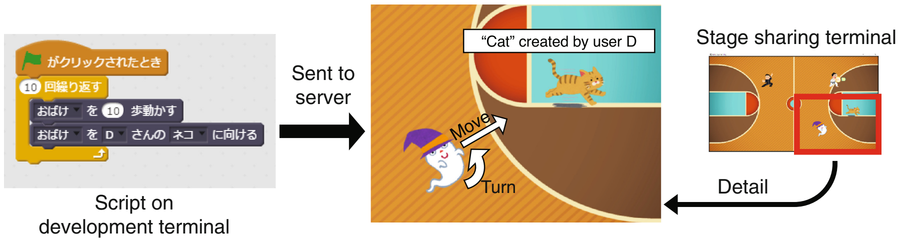


Fig. 11. Interaction between objects in group development

4 Application of the System

In this section we describe our experiences using the system for collaborative learning in practical programming workshops.

4.1 Overview

We held two programming workshops to evaluate our system, which we will refer to as ‘Workshop 1’ and ‘Workshop 2’. Table 2 shows the details of each workshop.

Table 2. Workshops

	Workshop 1	Workshop 2
Place:	Kodomo Mirai-kan (Kyoto City)	
Date:	November 27th, 2016	December 4th, 2016
Duration:	50 min	90 min
Participants:	4 (2 × 3rd + 2 × 4th grade)	7 (4 × 3rd, 1 × 4th, 1 × 6th grade, 1 × 1st grade junior high school)
Groups:	1	2 (3rd grade students, others)

We used normal laptop computers for the individual development terminals, and all-in-one desktop computers for the server and shared stage screen terminal.

Evaluation was performed during the workshops by observing the behavior of children while using the system, and after the workshops by conducting questionnaires.

4.2 Workshop Content

For these workshops we challenged the children to “make a dodgeball game”.

Participants created a game in which players throw a ball at each other on a dodgeball court displayed on the shared stage screen. In the first half of the workshops they worked individually to create a script moving an object with the arrow keys, and showing a ball at the position of the object when the space key was pressed. In the second half of workshops they changed from individual development to group development and modified their program to move the ball towards the objects created by other participants.

As they develop this kind of program in the workshop, children are expected to share information, teach each other new techniques, and improve their own programming by observing the different motions generated by the three solutions produced by the other group members. Creating a program that interacts with objects developed by the other participants promotes communication with them.

4.3 Results and Discussion

Tables 3, 4, 5 and 6 show the results of the questionnaires conducted after the workshops. Note that Q1 and Q2 in Workshop 2 (Table 4) are slightly modified from those in Workshop 1 (Table 3), reflecting the experience we gained during the first workshop. Q3 through Q6 were the same in both workshops.

Table 3. Questionnaire results for Q1 and Q2 in workshop 1

	Question	Votes
Q1	When did you look at projects of others? (single choice)	
	1. When I wanted to compare the movement of my project	1
	2. When I wanted to see the movement of others' project	3
	3. When I wanted to know the progress of my project	0
Q2	What did you teach or learn from others? (multiple choice)	
	1. I taught others how to use blocks and choose sketches	2
	2. I taught others how to move objects in a peculiar way	0
	3. I learned how to use blocks and choose sketches	3
	4. I learned how to move objects in a peculiar way	0
	5. Other (free description)	0

Table 4. Questionnaire results for Q1 and Q2 in workshop 2

	Question	Votes
Q1	When did you look at projects of others? (multiple choice)	
	1. When I wanted to compare the movement of my project	2
	2. When I wanted to see the movement of others' project	4
	3. When I wanted to know the progress of my project	2
Q2-1	What did you teach others? (multiple choice)	
	1. I taught others how to use blocks and choose sketches	1
	2. I taught others how to move objects in a peculiar way	2
	3. I didn't teach anything	3
	4. Other (free description)	
	• How to use the sound blocks	1
Q2-2	(If you chose 1, 2 or 4 in Q2-1) Why did you teach? (single choice)	
	1. I was asked a question	1
	2. Others looked troubled when I saw the movement of their project	1
	3. Others looked troubled when I saw the screen of their PC	0
	4. Other (free description)	
	• I wanted to teach	1
	• Others looked misunderstood	1
Q2-3	What did you learn from others? (multiple choice)	
	1. I learned how to use blocks and choose sketches	3
	2. I learned how to move objects in a peculiar way	3
	3. I didn't learn anything from others	3
	4. Other (free description)	0

Table 5. Questionnaire results for Q3 and Q4 in workshop 1 and 2

	Question	Votes
Q3	What was helpful to see the project of others? (multiple choice)	
	1. I used the same sketch because I liked it	2
	2. I liked the movement I didn't imagine	2
	3. I liked the movement with different ways of using blocks	4
	4. I didn't have anything helpful	4
Q4	Do you find anything that you wanted to imitate by seeing the project of others? (single choice)	
	1. I found something to imitate, so I created it by myself	3
	2. I found something to imitate, so I created it by learning from others	4
	3. I found something to imitate, but I couldn't create it	1
	4. I didn't find anything to imitate	3

Table 6. Questionnaire results for Q5 and Q6 in workshop 1 and 2

Q5	What did you think in creating a project with your team mates?
	I tried to make my project less overlapped with others
Q6	Write your impressions of this workshop
	• It was fun because I saw the way of programming of others.
	• It was fun!
	• I learned that I could create a game by programming with others
	• I was happy because I enjoyed Scratch very much and the workshop went well
	• Making a dodgeball game was great
	• I thought it difficult because this is my first experience, but it was good to go well

We will discuss the effectiveness of the system in two major areas, according to the results of the questionnaires and the behavior of the participants observed in the recorded video.

Communication and Co-teaching. As shown in the results of Q2 (Table 3) and with Q2-1 and Q2-3 (Table 4), most of the participants answered that they taught or learned from others during the workshop. We can conclude that active co-teaching occurred during the workshop. We can further conclude that the use of a shared stage triggered co-teaching because of answers similar to, "others looked troubled when I saw the motion of their project" given as reasons why participants felt motivated to teach. Sharing of the stage also generated opportunities to learn from others, as indicated by eight out of eleven participants who answered that, "I found something to imitate"; furthermore, four of these eight participants indicated that they created their programs by learning from others (Table 5).

The recorded video showed that communication among participants increased after the transition from individual development to group development. By collaborating in groups to create a single project, communication within the group increased. In the questionnaires, one participant answered Q5 by saying, "I tried to make my project less overlapped with others" (Table 6). The recorded video also showed that they tried to choose a different dodgeball target object from the other participants, by communicating among themselves.

Improvement of Programs. In response to Q1 ("When did you look at projects of others?"), there is a relatively large number of participants who answered, "When I wanted to compare the movement of my project" or, "When I wanted to see the movement of others' projects". From this we conclude that the shared stage increases the probability of participants referring to the projects and programs developed by others. From Q3 we also see that modification and improvement of programs was actually undertaken. Answers to Q6 similar to, "It was fun because I saw the way of programming of others" support the conclusion that participants were aware of what others were developing during the workshop.

From the discussion above, we conclude that the system is effective for promoting communication and co-teaching, as well as motivating participants to improve their own programs.

5 Conclusion

We described a collaborative learning support system based on Scratch that enables four users to share their stages on a single, shared screen. When the system was used during workshops we observed an awareness of others' projects during development, and communication and co-teaching being triggered by viewing the shared stage screen. It was also observed that sharing the stage and communicating with others led participants to make improvements to their own programs.

The current system limits the number of grouped terminals to four, because the shared stage screen is divided into four areas by the frame. In future we would like to make the system more flexible, for example by making it possible to divide the shared stage screen into six areas allowing six users to engage in group project work.

Acknowledgements. This work was supported by JSPS KAKENHI Grant Number 16H02925.

References

1. ScratchX. http://scratchx.org/. Accessed 21 Mar 2017
2. The Vision for ICT in Education, Ministry of Education, Culture, Sports, Science and Technology (2011). http://www.mext.go.jp/b_menu/houdou/23/04/__icsFiles/afieldfile/2012/08/03/1305484_14_1.pdf. Accessed 21 Mar 2017
3. Japan Revitalization Strategy 2016, Ministry of Education Culture Sports Science and Technology (2016). http://www.kantei.go.jp/jp/singi/keizaisaisei/pdf/2016_zentaihombun_en.pdf. Accessed 21 Mar 2017

4. Engelhard, P., Hirschfeld, R., Lincke, J.: Pitsupai collaborative scripting in a distributed, persistent 3D world. In: Proceedings of the Seventh International Conference on Creating, Connecting and Collaborating through Computing, pp. 87–94. IEEE (2009)
5. Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E.: The scratch programming language and environment. ACM Trans. Comput. Educ. (TOCE) **10**(4), 16 (2010)
6. Stewart, J., Bederson, B.B., Druin, A.: Single display groupware: a model for co-present collaboration. In: Proceedings of the SIGCHI conference on Human Factors in Computing Systems, pp. 286–293. ACM (1999)
7. Takada, H.: A 3D collaborative creation environment with tile programming on croquet. In: Proceedings of the Fifth International Conference on Creating, Connecting and Collaborating through Computing, pp. 125–130. IEEE (2007)
8. Umezawa, M., Abe, K., Nishihara, S., Kurihara, T.: NetMorph-an intuitive mobile object system. In: Proceedings of the First International Conference on Creating, Connecting and Collaborating Through Computing, pp. 32–39. IEEE (2003)