Algoblock: a tangible programming language, a tool for collaborative learning

Hideyuki Suzuki
Ibaraki University
34 PUBLICATIONS 263 CITATIONS

SEE PROFILE

Hiroshi Kato
The Open University of Japan
64 PUBLICATIONS 352 CITATIONS

SEE PROFILE

AlgoBlock: a Tangible Programming Language - a Tool for Collaborative Learning

Hideyuki Suzuki, Ibaraki University
hideyuki@suzuki-lab.net
Hiroshi Kato, Open University of Japan
hiroshi@kato.com

1. Programming languages as conversational artifacts

The authors intend to design a learning environment in which learners at primary and secondary school can improve their skills for planning and building procedures through group work. Group work is essential for learning from the standpoint of situated learning theory [1][3] which considers that learning is a process of enculturation through social interactions.

In designing the learning environment, although social setting for authentic activities [l] is important , it is also important to design the tools for activities, because the structure of our activities is affected by the constraints of the tools in use. Here, the authors focus on the programming languages as a conversational artifact [4][5] which facilitates interactions among learners.

We believe that programming languages for collaborative learning ought to be designed according to the following principles: a) easy operation: Learners can operate the system easily. Easy operation contributes deeply to the promotion of collaborative activities for the following two reasons. 1) learners do not have to concentrate on how to use the tool, consequently they can be absorbed in group work, e. g., seeking solutions through discussion, building procedures and examining them collaboratively. 2) easy operation promotes trial-and-error activities, which draw conversations among learners. b) simultaneous accessing and c) mutual monitoring: all learners are equally eligible to refer to and manipulate the program whenever they want to. And all learners can monitor the process of the collaborative activity mutually. These features contribute to the activation of collaboration, because they promote learners' activities, such as displaying their own ideas to the others when they want to, understanding the others' operation to the program immediately, and intervening against the others' operations on the spot. d) natural tum-taking control: tum-taking to refer to and manipulate the program should be controlled, since unlimited simultaneous accessing to the representation by all participants inevitably brings out confusion and breakdown of the collaboration. Rules for acquiring control should be natural for learners, because obeying unusual rules prevent learners from collaborating smoothly.

Preceding research has pointed out the possibility of programming languages including Logo as a conversational tool. (e. g., [6]) However, those systems have innate limitations as a conversational tool. The limitations are mainly from constraints on the human-interface of the systems. Generally, the systems use the CRT as a display device; however, only a few viewers can face a CRT, because it is not designed to be shared by two or more users. Therefore, all learners cannot refer to the display simultaneously in collaborative work. This prevents learners from monitoring the process of the works. In addition, those systems employ keyboard or mouse, which allows only one learner at a time to use them to manipulate the programs on the CRT display. Thus, all learners cannot manipulate a program at the same time. Moreover, with these input devices, the tum-taking is controlled by handing over the devices among learners. This forces learners to obey an unnatural rule of tum-taking that is significantly different from the rule of everyday collaborative activity in which tum-taking is unconsciously controlled mainly by non-verbal cues. This limitation can be an obstacle to learners' interaction. Considering these limitations mentioned above, the existing programming languages are not completely suitable for collaborative learning. Based on these investigations, AlgoBlock was developed.

2. What is AlgoBlock ?

AlgoBlock (Fig.1), a Logo-like tangible programming language, is a set of physical blocks that can be connected to each other manually to form a program. Those blocks are then connected to a computer to execute the program. Each block corresponds to a command of a programming language. Learners can operate a graphically displayed submarine on the CRT (Fig.2) through Logo-like commands, such as go forward, slide to left/right, turn, unfold the legs for landing, and so on. Each one of those commands has a corresponding physical block. In addition, control blocks, i.e., if-then-else and repeat -until are available for logical control. Some kinds of blocks have a parameter switch on them. The switch is used to set a parameter for a command. Learners make a program by connecting real blocks and setting parameter switches. The purpose of AlgoBlock is to allow learners in primary and secondary school to improve their skills for planning and building procedures through group activities.

==== insert Fig. 1 and Fig. 2 here ====

3. AlgoBlock as a conversational artifact

AlgoBlock has been designed as a tool for collaborative learning according to the following principles which deeply support conversation. In other words, it is qualified as a conversational artifact.

a. Easy operation

With AlgoBlock, learners can make programs by handling tangible blocks. This way of making programs is much more direct, therefore, easier than inputting the program with a key board and an editor. Easy operation evolves learners in collaborative work, and promotes trial-and-error activities which generate conversation.

b. Simultaneous accessing

Learner's intentions are physically represented on the blocks, and placed on the space of collaborative activities. This tangible representation allows simultaneous pointing and manipulation by a group of learners, so all learners are equally eligible for accessing the program; i. e., AlgoBlock enables learners to share the representations of an algorithm.

c. Mutual monitoring

Manipulating AlgoBlock generates apparent gestures. Toe gestures indicate their intentions externally. This enables all learners to monitor each other's operations on the blocks throughout the collaborative work. The open interaction stimulates conversation among learners and facilitates collaboration. In other words, AlgoBlock works as an open tool [2], which enables users to monitor their cooperative activities.

Moreover, AlgoBlock shows the program execution dynamically. Each block has a lamp that flashes when the corresponding command is executed. Indication lamps enable learners to trace the flow of program. In addition, AlgoBlock has a debug mode in which learners can trace the program step by step, by just pushing a button on each block. These functions also help learners to monitor the process of activities.

d. Gesture based turn-taking control

As mentioned above, operating AlgoBlock naturally generates apparent gestures (Fig.3). For example, stretching out one's hand, turning one's body towards the blocks, and other gestures that are inevitably generated in the natural flow of operations on the tangible tool. Consequently , tum-taking of access is controlled by those gestures; e. g., learners get turns and interrupt others' operations by stretching their hands toward the blocks. This sort of control contributes to facilitation of collaborative activities. This is because, rules for acquiring control are equally the same as in everyday activities, thus learners never need to learn artificial and system specific rules. As a result, learners can cooperate unconsciously .

=== insert Fig. 3 here ===

4. Observation

One experimental session in which three 12-year-old female children were engaged in a group work with AlgoBlockfor one hour was executed. The purpose of the session is to examine the effect of AlgoB!ock on collaborative work. All subjects had no preceding experience in programming or manipulating computer based tools.

In the session, 3 tasks in which the subjects were required to move the submarine towards the goal were given after a 3-rninutes introductory phase. In the introductory phase, they were instructed to observe the work with AlgoBlock which was executed by experimenters for a while.

Although two experimenters attended the session, they never intervened with the subjects' work with the exception of system trouble solution. Subjects' conversation and non-verbal actions were recorded throughout the session. Following is the result of the qualitative analysis of the protocol data.

Throughout the session, it is observed that the subjects made and changed programs easily. Although, it was the first experience for all subjects in manipulating programming language, they could operate the language without difficulty after about 3 minutes of the introduction phase. The following protocols show that AlgoBlock is easy to operate. All utterances in the protocols are originally in Japanese.

<notations>
CRT: description of CRT display (program execution)
Subx : Subject identification index
{ }: non-linguistic action
( ): Authors' comment
[ ]: Index for explanation

<protocol 1>
CRT : End of execution (submarine failed to reach the goal)
Sub2 : " This is wrong. " { assigned Right45 to the ROTATE block [action-1] J

<protocol 2>

CRT : End of execution (submarine collided with an obstacle)

Sub3 : " This { pointed a block} might be Left45. This direction. " {moved hand form the upper right to the lower left l

Sub 1: " I see. " { changed a ROTATE block's parameter into Left45 [action-2]}

Subjects changed the program immediately after observing the execution. (e. g., [action-1], [action-2]) Easy operation, which is mainly from tangibleness of the tool, encourages learners' trial-and-error activities. Vigorous trial-and-error activities in collaborative work promote generating topics of conversation, thus the collaboration is activated.

<protocol 3>

Subl : " Hey. Is 2 okay? { pointed the parameter switch of the FORWARD block} Sub2 : " Yah, go forward 2 steps. "

Sub3 : " 2 steps forward, I think so. { moved a FORWARD block to front of Sub2 } Then? " Sub2 : " To right. "

Sub 1 : " Rotate, rotate 90 degrees. "

Sub3 : { connected the SLIDE-RIGHT block to one side of the FORWARD block}

Subl : " How many steps to right? "

Sub2 : " 2 steps! "

Sub 1 : " Then, rotate. "

Sub2 : " Rotate 45 degrees. "

Sub l : {connected the ROTATE block }

Sub2 : " And then, one, two, three . 3 steps. " (connected the FORWARD block (3 is assigned as the parameter) I

From protocol 3, it is observed that they collaborated by sharing a temporary and incomplete representation which was placed on the space of collaborative activities. They never built whole plans in their mind s, but they refined their plans incrementally through conversations. In addition, all the subjects were equally eligible to manipulate and refer the tangible representation in the activities.

Protocol 4

Sub2[A]: " Has folded the legs (legs for landing), now it (the submarine) can slide. (boosters for sliding right and left are available only when the legs are folded)" {pointed the LEG-FOLDING block [action-a l] }

Sub 1[B]: “ These two must be reversed." { turned body towards the blocks [action-b l] , and pointed the LEG-FOLDING block and the ROTATE block [action-b2] }

Sub2[C]: " No. This way. " {tried to change the parameter for the FORWARD block [action-cl], and prevented Subl 's pointing (against [action-b2]) by stretching elbow }

Sub l [D]: " No! First, we must make the legs fold, and then rotate 90 degrees. " {shouted, leaned towards the blocks [action-d l], and pointed the ROTATE block [action-d2]}

Sub3[E]: " I think so. "

Sub 1[F]: " Because the legs aren't folded now. " {tried to make the blocks, i. e., LEG-FOLDING block and. the ROTATE block, reversed [action-fl]}

Sub2[G]: " No. This must be (not clear) "

Subl[H]: " I'm certain." {made the blocks reversed [action-h l] }

Manipulations on AlgoBlock generate apparent gestures, e. g., [action-c 1], [action-f 1].This tendency might be much more noticeable than the situation in which the subjects use the keyboard or mouse for input. Utilization of tangible tools enable all subjects to monitor the collaborative work. For example, Sub 1's [F] operation on the blocks are indicated clearly as obvious gesture [action-fl] to al I subjects. Sub2 grasped the intention of Subl from Subl 's action, and made objection to the operation [G] immediately. This open interactions stimulated conversation among the subjects, and facilitated collaboration.

Protocol 4 also shows turn-taking to access the program was controlled with AlgoBlock. Subl [B] tried to manipulate the program by turning her body towards the block s and reaching out a hand to the blocks · [action-b1 / action-b2]. Against Sub 1, Sub2 prevented the action by swishing away Sub 1'shand [action-c2]. After that, Sub I leaned towards the blocks again [action-dl /action-d2], and finally succeeded in operating the program [action-hl]. Here, the tum was shifted by the subjects' actions on the program which are generated naturally in the activity .This gesture-based control facilitates collaboration by reasons mentioned in the preceding section.

Moreover, it was observed that the subjects used abridged expressions frequently. For instance, Sub l [B] said "These two must be reversed" by pointing the LEG-FOLDING block and the ROTATE block directly [action-b2], instead of saying "How about reversing this LEG-FOLDING block and that ROTATE block". In this way, they expressed their intentions immediately and easily, which supported learners' conversation. These abridged expressions make sense when they set consciousness onto the same objects. AlgoBlock supports learners in the sharing of the orientation of consciousness because of its tangibleness which provides apparent targets of mutual consciousness-setting. In other words, AigoBlock has competence to generate collaborative space successfully.

5. Conclusion

In the preceding chapters, the role of programming languages as conversational artifacts was emphasized, and requirements for conversational support were examined. Based on this investigation, the authors proposed the concept of AlgoBlock which is a tangible programming language for children, and mentioned its aspect as a conversational artifact. Results of observations were also presented. Although it is incomplete and more analysis is needed, the results show that AlgoBlock is qualified as a conversational artifact.

Although it turned out that AlgoBlock is an adequate tool for conversation, it is not the only solution for conversational support. The authors intend to propose principles for designing collaborative learning environment from aspect of tools as well as social settings. In this perspective, AlgoBlock is a first step of our research.

REFERENCES

[l]Brown, J. S., Collins, A. and Duguid, P. (1988):"Situated Cognition and the Culture of Learning", IRL Report No. IRL88-0008

[2] Hutchins, E. (1990): "The Technology of Team Navigation" , Intellectual Teamwork, Galegher, Kraut and Egido (Eds.) pp.191-220, Erlbaum

[3] Lave, J., and Wenger, E. (1991):"Situated learning - Legitimate peripheral participation -" Cambridge University Press

[4] Pea, D.R., Sipusic, M. and Allen, S.(1990):"Seeing the light on optics: classroom- based research and development of a learning environment for conceptual change", Proceedings of 7th Tel Aviv Workshop on Human Development: Development and Learning

[5] Roschelle, J . (1990): "Designing for conversations", AAAI Symposium on Knowledge-Based Environments for Learning and Teaching (1990)

[6] Schibeci, R. A. (1988):"Logo in preservice and inservice teacher education", Proceedings of 6th Australian Computers In Education Conference

Figures

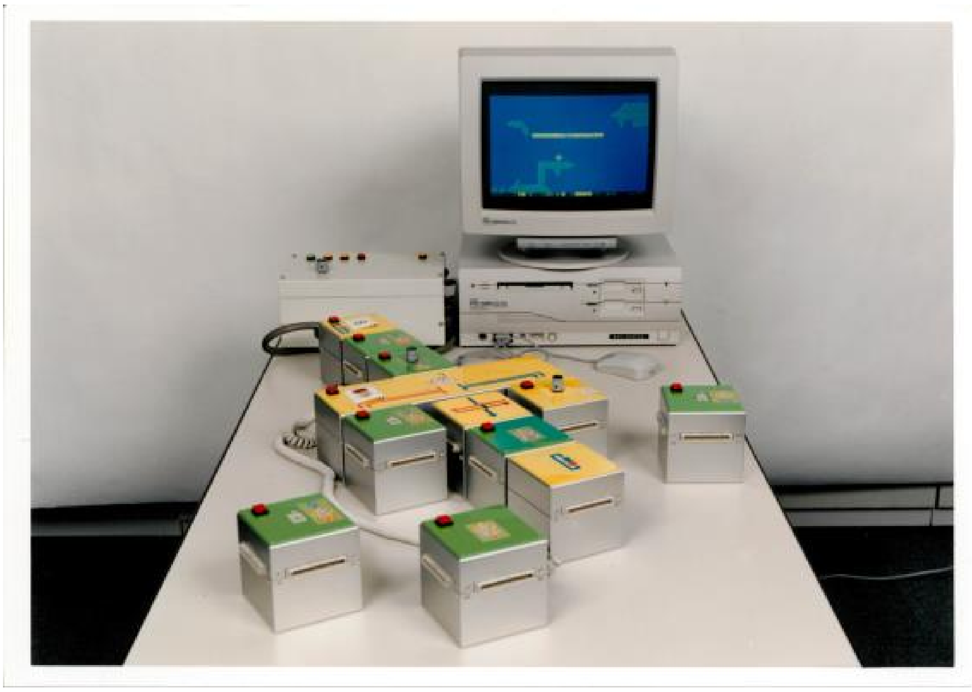

Figure 1: A tangible .programming language: AlgoBlock

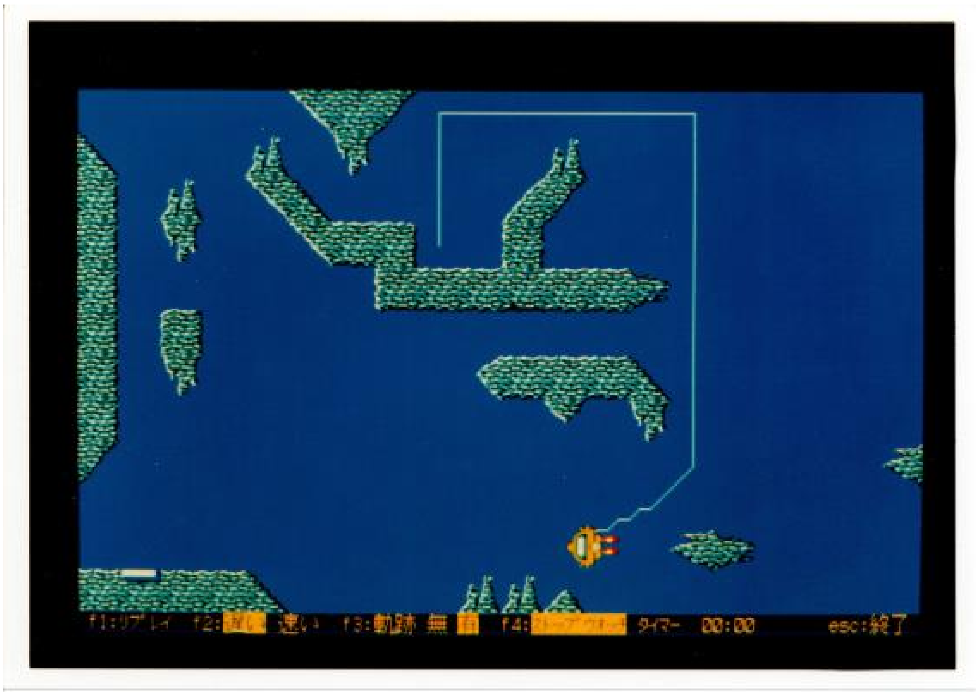

Figure2: Screen Design of AlgoBlock

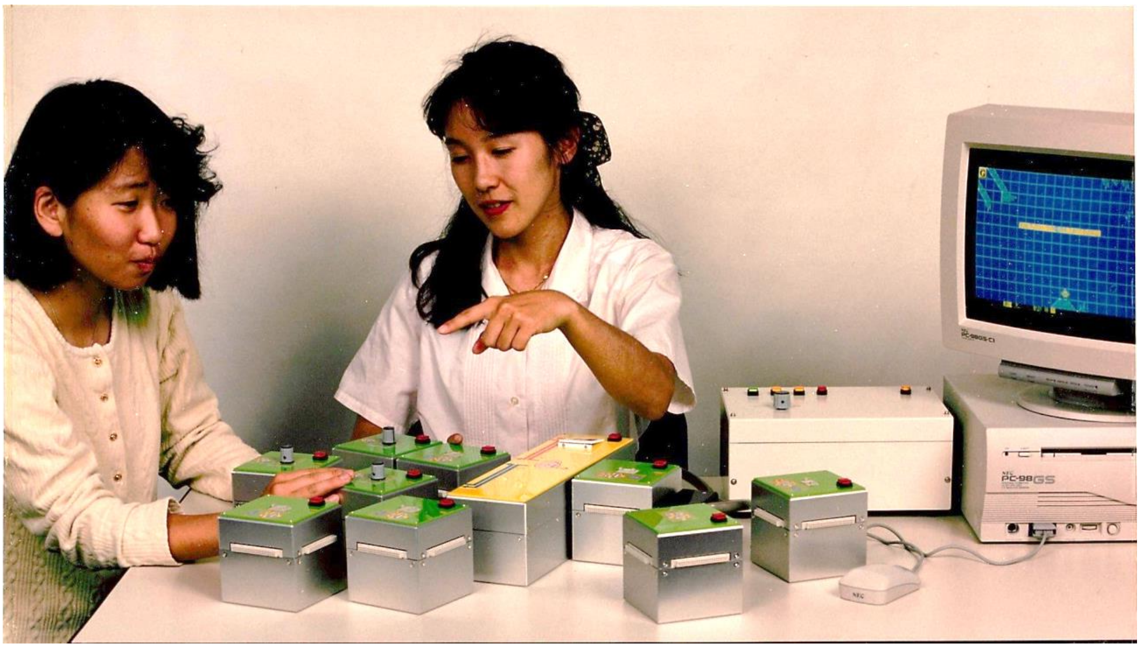

Figure 3: Collaborative work with AlgoBlock (adult)