CoProStory: A Tangible Programming Tool for Children's Collaboration

Xiaozhou Deng, Danli Wang, and Qiao Jin
dengxiaozhou2018@ ia.ac.cn, danli.wang@ia.ac.cn, jinqiao2018@ia.ac.cn
The State Key Laboratory of Management and Control for Complex Systems,
Institute of Automation, Chinese Academy of Sciences, Beijing, China
School of Computer and Control
Engineering, University of Chinese Academy of Sciences, Beijing, China

ABSTRACT: In this paper, we present CoProStory, a collaborative tangible programming tool to support children learning programming. In recent years, many collaborative tangible programming tools were developed for children. Compared to the other tools, Sync Blocks which were designed to coordinate children's programming were used in our tool. The tool consists of two parts: tangible programming blocks and game tasks. Game tasks are provided with 3D cartoon story scenes on computer screen and need to be finished by collaborative programming with our programming blocks. To evaluate our tool, a contrast user study was conducted with 28 children aged 6 to 10. The results demonstrated that CoProStory could support children to program collaboratively and children had a more positive attitude toward programming with Sync Blocks.

Introduction

Children programming has been a wide field of research since 1960s. Research has shown that programming education can have a positive and measurable effect on children's achievement, not only in areas such as math and science, but also in language skills, creativity, and social emotional interaction (Douglas, 1999). Besides, learning programming is an efficient way to cultivate computational thinking which has been described as a fundamental skill for everyone, not only for computer scientists (Jeannette, 2006).

However, programming appears to be quite a challenge for young children. With the traditional way of programming, children have difficulties not only in learning rigid syntax and text symbols, but also in using the complex programming environment (Andy & Andrew, 2007). Therefore, lowering the difficulty of programming and offering children proper programming tools are highly valuable in education. TUI is an efficient interaction method embracing the richness of human’s interaction with the physical world, which contributes to children’s learning (Michael & Robert, 2007). With tangible programming tools, children can write programs by assembling the physical objects without keystrokes, which is much easier to involve children in programming (Michael & R.Jordan et.al, 2012; Timothy, 2004). Besides, TUI has the advantages of involving multiple children in the same process, as TUI can easily provide an open and shared environment among children (Hiroshi & Brygg, 1997).

Face-to-face collaboration with classmates or friends is an important part of children's daily lives. Collaboration provides children with an efficient way to improve social communication skills. It helps children analyze information from the perspectives of other users when communicating face-to-face (Regan & Kori et.al, 2001). Collaborative programming can promote the awareness of the teamwork, and the skills of communicating and collaborating (Scott, Regan & Kori, 2003). However, for children, if the tasks are not assigned well, the inequality of task loads can result in significant decrease of children's practice time, interests and concentrations. And, research has indicated that tangible user interfaces could provide broad and shared interaction environments, which have the potential to involve multiple children at the same time and support face-to-face collaboration (Leslie, 2007). Furthermore, research implicated that collaborative technology should support concurrent interaction which can help to engage children in a collaborative activity and enable them to participate equally (Neha & Laurie et.al, 2004; Charlie & Linda et.al, 2002).

Based on the analysis above, we applied the concept of process synchronization and developed CoProStory, a tangible programming system supporting children's collaboration in programming (Figure 1). Compared with previous works, CoProStory allows two children to program their own characters with programming blocks in parallel, which aims to complete a common task. In this method, with clear role division, conflicts could be well reduced. Furthermore, Sync Blocks are provided to coordinate children's collaboration.

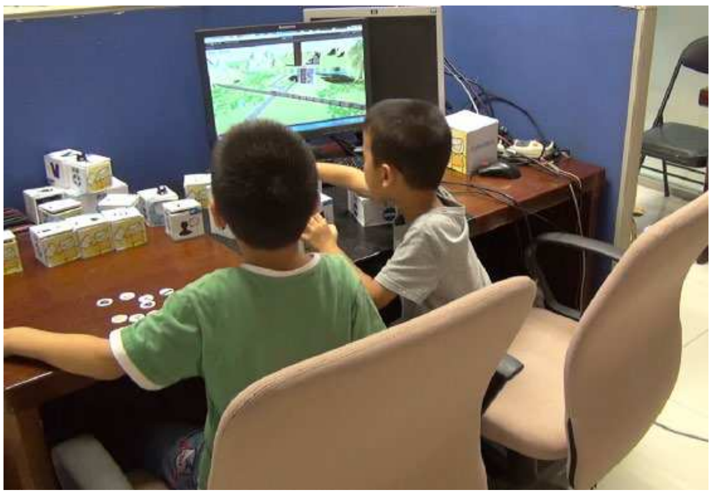
Figure 1. Children playing with CoProStory.

Related work

One of the earliest tangible programming projects that support collaboration is AlgoBlock (Hideyuki & Hiroshi, 1995). Children write their own programs to play a marine game by connecting the objects. In their user study, the authors proved AlgoBlock's value in improving the communication among children by recording and analyzing their speech and activities. Tangicons 3.0 is an educational game designed for children age 6 - 9 (Florian & Thomas et.al, 2012). It provides a collaboration environment by allowing at most four children to program together to move virtual characters on a map. However, its programming blocks are built based on Sifteo cubes, which increase the cost of the tool. Digital Dream Lab (DDL) is a tangible storytelling system for children to learn programming concepts. DDL focused on the concepts of variables and classes by using the metaphor provided by irregular puzzle pieces (Hyunjoo & Anisha, 2013). The system has six kinds of blocks, including Character Block, Animation Block, Color Block, Size Block, Variable Block and Background Block. However, the tool was also designed for older children (7-12 years old) and younger children are short in collaboration compared to those of older children. Plugramming was designed to support young children's face-to-face collaborative programming. However, in this tool tasks are divided into two separated branches of a program (Tomohito & Yasushi et.al, 2017).

From the works above, we find the open characteristics of TUIs can effectively support face-to-face collaboration between children. Based on previous work, we designed and developed a new tangible programming system, CoProStory, which allows two children play and program face-to-face. With clear role division and Sync Blocks, CoProStory could support two children to program concurrently and collaboratively.

Design and implement

Our design and implement process involved two major revisions in which we tried to explore how to encourage children's collaboration better. The difference between the two versions is that one is designed to be completed with Sync Blocks while the other isn't. The details are described as follows:

Programming blocks

Programming blocks are small square boxes (6cm*6cm*6cm) covered with color figures representing certain semantics or manipulation constraints (Figure 2). Programming blocks are the primary tools for children interacting with the animation game running on the computer.

Blocks with RFID reader are very flexible, and are suitable for parameters with a lot of optional values. In these blocks, a RFID reader was fixed beneath the top surface. And with a shallow groove on its top surface, a mini Interactive Element Card (IECard) could be embedded, together making up a complete semantic. IECards are made up with small circular RFID tags covered with figures, which could represent different interactive elements in game scenes, such as a tree, a gate and even a character. And with infrared sensor fixed inside, the message of related position could be provided.

Programming blocks can be divided into four categories: Character Blocks, Attribute Blocks, Animation Blocks and Sync Blocks. Children could use Character Blocks and Attribute Blocks to initialize their characters, which provides with clear role division. Since Attribute Blocks provide children with individual changes to visual character, it may be easier to make children immerse themselves to our game. Animation Blocks are used as commands to program the characters' actions. With different kinds of IECards, children could use Animation Blocks to program the visual character finishing the given tasks. Most importantly, we design a special type of blocks, Sync Blocks, to coordinate children's collaboration.

Sync Blocks are designed to encourage children's collaboration in the game. Three kinds of Sync Blocks are currently provided: Wait Blocks, Notice Blocks and WaitForSeconds Blocks. Wait Blocks and Notice Blocks are always used in our programs. Once it comes to a wait command, the corresponding character would stand there and keep waiting until another character comes to a notice command. To use the Sync Blocks properly, children need to decompose the given task together before programming and figure out a solution with provided blocks.

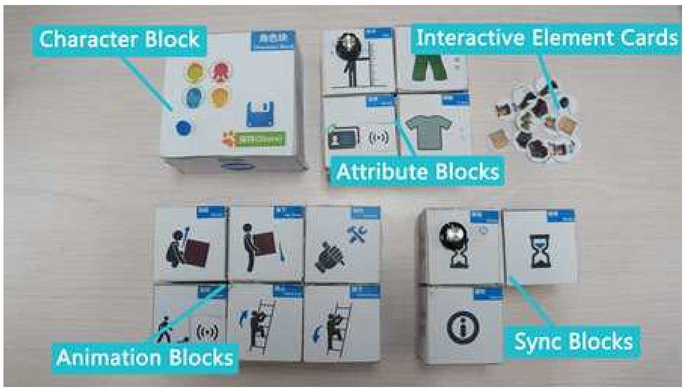

Figure 2. Programming Blocks & Interactive Element Cards.

Sync Blocks help children to keep in the same speed to some extent. If children ignore their partner's process, the real-time feedback would provide corresponding error messages. For instant, one character needs to cross a river but he should wait for the other to drop the suspension bridge and children should use Sync Blocks to complete these moves.

Visual games

We provide two versions of games which need different degrees of collaboration. Specifically, the work in the first version is designed without Sync Blocks.

In the game of version1, children should use programming blocks to move their characters to the given destination. One of characters needs to go to the tree from the house while the others direction is opposite. During this process they should make their characters finish certain movements such as opening an umbrella or striding over a rock in order. They need to program the correct sequences without Sync Blocks.

In the game of version2, children need to finish their task with Sync Blocks. After several movements, one character should wait for the other dropping down the suspension bridge by a series of operations to cross the river. After the character got cross the river, the other character should lift the suspension bridge again. In this story scene, two characters' movements are time-related, as a result, children need to use Sync Blocks to finish this given task.

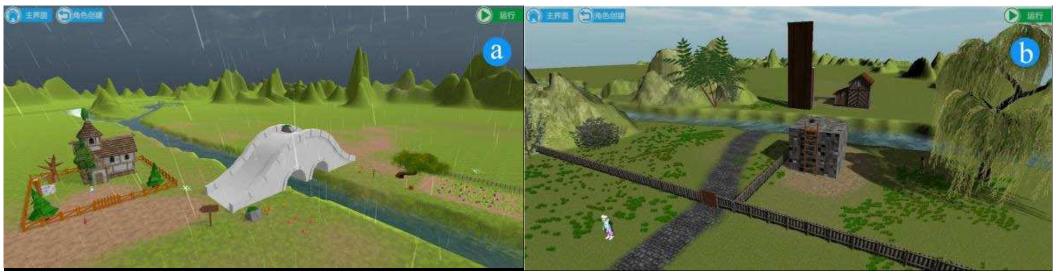

Figure 3. Different versions of game tasks.

Programming process
The whole process of learning consists of three stages: initialization stage, programming stage and running stage. Children would use Character Blocks and Attribute Blocks to initialize their characters. Then simple background stories would be told to children which contains game task. After children get clear with their goals, they are asked to program characters to complete given tasks. During this programming stage, the real time feedback showed in figure 4 would help children and reduce their cognitive load of programming. Specifically, there would be states messages on the screen which could show children their progress in real time. And if children place wrong blocks, there would be error messages to help children debug their program. After children finish their program, the running state would show the result of children's programming with animation. We use these states to introduce children the normal process of programming.

To evaluate our tool, we conducted a contrast user study and compare these two versions to find the benefits of Sync Blocks.

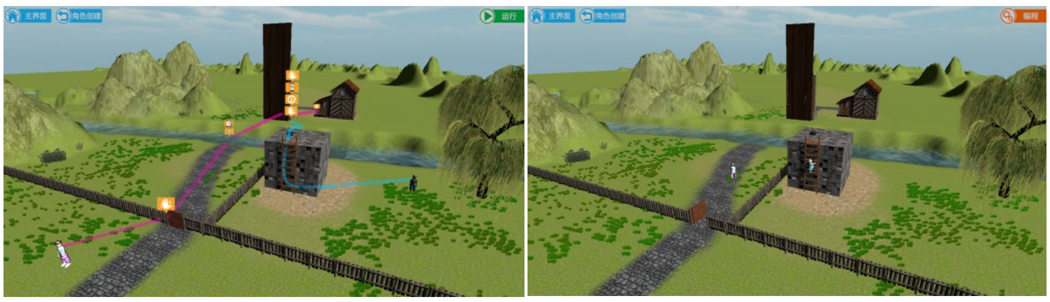

Figure 4. Programming state and running state.

User study

Participants and setting
28 children (13 girls and 15 boys) aged 6 to 10 with a mean age of 7.79 were involved in our user study. To assess the collaborative design of CoProStory, 28 children were divided into two groups evenly: G1 (7 girls and 7 boys with mean age of 7.5) and G2 (6 girls and 8 boys with mean age of 8.14). Children in G1 were experimented with Version1 of CoProStory and children in G2 experimented with Version2. Experiments were conducted in a spacious room. In order to capture children's behaviors and voice, a video recorder was set up.

Experimental procedure
The experimental procedure mainly consists of three parts: introduction, programming and interview. At the beginning, we introduced CoProStory to children (5 minutes). After that, children were asked to complete the programming tasks in pairs (30 minutes) and the interactions, conversations, utterances and time they spent on each task would be recorded with cameras. Next, we made a brief interview for children (10 minutes). Altogether, one experiment takes one regular school lesson (45 minutes).

Measures
In ordered to find the advantages of the design of Sync Blocks with quantitative data, we conducted quantitative coding and analysis of video data of the participants' behaviors act by act using Bale's Interaction Process Analysis coding scheme (Robert, 1950). And in order to raise measurement accuracy, each video data would be analyzed by two coders. Firstly, the two coders would learn the coding schema and coding rules, then coded 2 groups of video data independently. Afterward, they discussed their disagreements, adjusted their mismatched codes and then coded the remaining data separately.

According to Bale's Interaction Process Analysis coding scheme, we used these four kinds of codes to evaluate the interaction of children while programming: 1) positive reactions, 2) negative reactions, 3) initiative helping behaviors, 4) behaviors of asking for assistance. With the quantitative data of these codes, we could measure the collaboration of children and compare the results of the two groups to find the positive effects of Sync Blocks. Table1 is the description of each code.

Table 1: Coding scheme used for our analysis of the recorded video

Code	Description
C 1	Positive Reaction, including 4 kinds of behaviors: task-related conversation, task-related respond, passing blocks and correct partner's errors.
C 2	Negative Reactions including refuse to help/answer and ignore partner's task-related talk and request.
C3	Take the initiative to provide help and answer task-related questions.
C4	Ask for help and ask task-related questions.

And we made a brief interview after children programmed. The interview questionnaire is a Likert-type scale composed of 8 questions scored from one to five, where one means the minimum and five means the max score. Q1-Q4 concerned about CoProStory's interests, ease to use and learn. Q5-Q8 are used to evaluate how CoProStory support children's collaboration. Questions are as follows:

- Q1: How much do you like the game?
- Q2: Do you think the game is easy to learn?
- Q3: Do you think it is easy to control roles in the game by blocks?
- Q4: Was the real-time feedback in the game helpful in the programming stage?
- Q5: How much do you like playing the game with your partner?
- Q6: Do you pay attention to your partner when you were playing the game?
- Q7: How much help did you offer your partner?
- Q8: How much help did your partner offer you?

Results

We analyzed the quantitative data of four codes combined with video data. The quantitative coding results are shown in Figure 5:

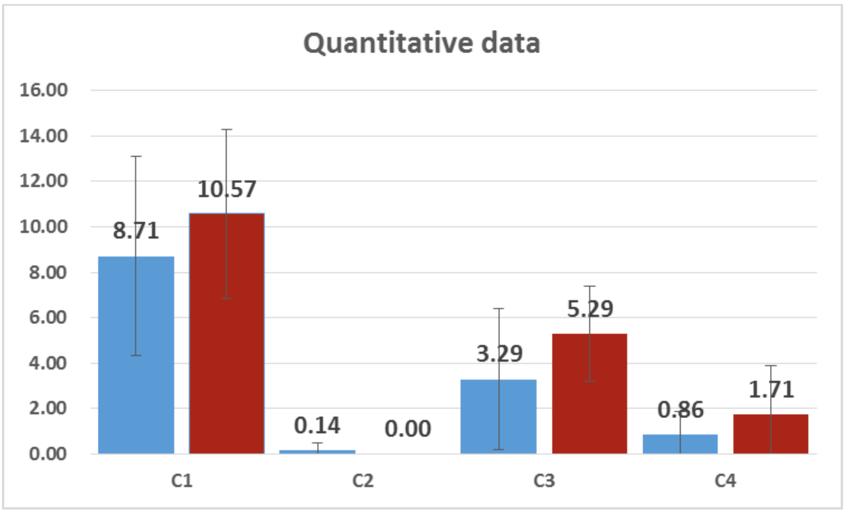

Figure 5. Quantitative coding results of G1 (blue) and G2 (red).

For interactive behaviors, the average positive data (C1) of G1 and G2 are 8.71(SD=4.40) and 10.57(SD=3.74), which indicates that CoProStory's design affects participants' collaborative behaviors and children would show more positive reactions in version 2. The average negative data (C2) are 0.14(SD=0.35) and 0(SD=0). The negative data are both low, which indicates that children would unlikely show negative reactions while programming with CoProStory.

For helping behaviors (C3), children who used G1 showed relatively fewer helping behaviors (M=3.29, SD=3.1) compared to children who used G2 (M=5.29, SD=2.12). And for C4, children in G2 were more likely to ask for help (M=0.86, SD=0.99) than G1 (M=1.71, SD=2.19). The results of G2 were relatively higher than G1 which means the design of Sync Blocks can promote children's helping behaviors. Combined video data, we can find that children are more likely to help their partners while programming with Sync Blocks. Here is an example in G2:

Children1: Now I need the suspension bridge down to continue. Using Wait Blocks to wait...where do you get? I need you place the Notice Blocks.

Children2: I haven't got there. And I need the climb Animation Block.

Children1: Here is your Block. What's next?

Children2: Let's see...

As the example shows, children in G2 were more likely to pay attention to their partners with Sync Blocks while programming. By contrast, according to the video data, children in G1 often offered initiative help until they finished their own tasks. Here is an example in G1:

Children1: I have finished my part!

Children2: I still need...

Children1: It's easy. Let me help you.

The results of our interview are shown in Figure 6. Overall, scores of interviews show that CoProStory could support children programming collaboratively.

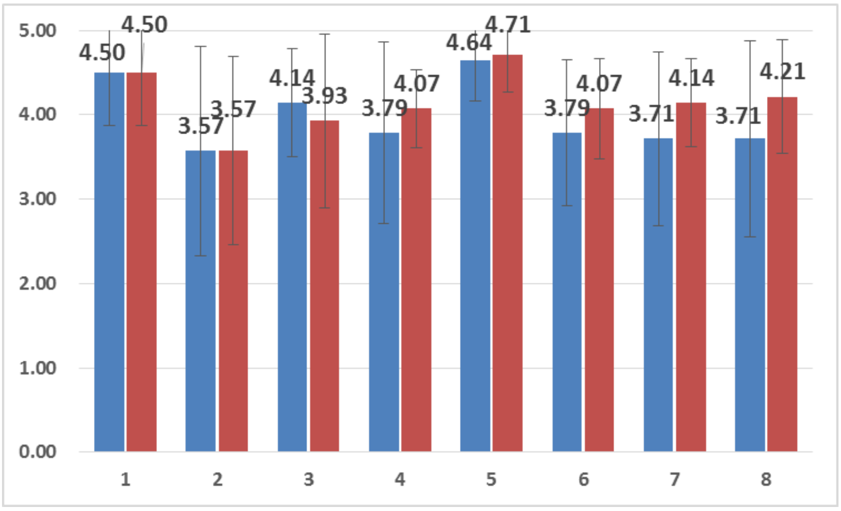

Figure 6. Scores of Q1-Q8 (blue columns are scores of G1, red columns are scores of G2).

The result of the interview shows that CoProStory is interesting, easy to use for children for the reason that children in both G1 and G2 gave CoProStory average 4.5(SD=0.63). But the scores of Q2, both G1 and G2 result in 3.57, indicate that we may make the semantics of the blocks clearer to reduce the cognitive load. As for Q3, the average scores (4.14 with SD=0.64 and 3.93 with SD=1.03) of two groups were positive which means children thought it is easy to use programming blocks to control the characters. Q4 is about whether the real-time feedback in the game helpful for children in the programming stage. Children in different groups gave average scores of 3.79(SD=1.08) and 4.07(SD=0.46). The result shows a positive evaluation of CoProStory's real-time feedback and indicates that the real-time feedback was helpful for children to complete the tasks.

In Q5, children were asked how much they like playing the game with their partner, they gave the question average 4.64 (SD=0.48) and 4.71(SD=0.45). Both groups gave high scores. In Q6, we asked the children if they paid attention to their partner when they were playing the game. They gave the question average 3.79 (G1, SD=0.86) and 4.07(G2, SD=0.59). A relative higher score of G2 indicates that children may pay more attention to their partner with Sync Blocks. In Q7, we investigated how much help did they offered their partner, children gave an average of 3.71 (G1, SD=1.03) and 4.1(G2, SD=0.52). Children also gave an average of 3.71(G1, SD=1.16) and 4.21(G2, SD=0.67) for Q8-investigating how much help their partner offered them and the difference between G1 and G2 indicates that Sync Blocks may lead children help each other.

Discussion

CoProStory is a new tangible programming tool for children's collaboration and the design of Sync Blocks is to promote collaboration and avoid inequality of loads of programming. According to the results of user study, children were more engaging and showed more positive reactions while programming with Sync Blocks. But as the results of interview shows, there isn't significant difference in the scores of interests between these 2 versions and children gave lower scores of eases to learn in G2. The reason may be that the new Sync Blocks also increase the cognitive load of programming while coordinating children's programming.

To avoid inequality of loads, we make clear role division and assign tasks to children as expected. However, the separation may cause less collaboration and the difference of programming speed may still cause inequality, which is reflected in G1: children in G1 often offered initiative help until they finished their own tasks and as consequence the children got help might take less loads. To avoid inequality and promote collaboration, we use Sync Blocks to make the separated tasks time-related and coordinate children's programming. And based on our experiment, we think it's necessary to classify the help behaviors and find new measurement of inequality to get a better comparison. Besides, the design of visual games with Sync Blocks needs further consideration. For instance, the positions of time-related nodes effect children's collaboration and the method of nodes assignment needs further research.

Conclusion

This paper described CoProStory, a brand-new tangible programming tool which could encourage collaboration between children. The tool supports children to program together in the same physical environment while work on a shared quest. Besides, with the comparison of the two versions, we found the advantage of the design of Sync Blocks.

We try to applied the concept of process synchronization to promote children's collaboration. As described, we design Sync Blocks to coordinate children's programming and made a contrast user study to explore the better way to support children's collaboration better. And as the results of our experiment show, children were more positive in version 2. In particular, some children who program faster usually would help their partners if their program was blocked by Wait Blocks due to the turning of attention to their partners. Besides, children would observe other programming while using Sync Blocks, which usually indicates positive discussions.

In the future, this work could be improved in several ways. As the results of the interview show, we need to improve the interaction of CoProStory. For instance, we can make our real-time feedback clearer. Furthermore, the interplay between quality of collaboration and learning outcome is worth to be further studied in depth through designs of more specific learning modules and corresponding evaluation methods.

References

Andy Cockburn, Andrew Bryant. (1997). Leogo: An equal opportunity user interface for programming: Journal of Visual Languages & Computing 8, 5: 601-619.

Charlie McDowell, Linda Werner, Heather Bullock and Julian Fernald. (2002). The Effects of Pair-Programming on Performance in an Introductory Programming Course. In Proceedings of the SIGCSE technical symposium on Computer science education (SIGCSE'02), 38–42.

Douglas H. Clements. (1999). The future of educational computing research: the case of computer programming. Information Technology in Childhood Education Annual, 147-179.

Florian Scharf, Thomas Winkler, Claudia Hahn, Christian Wolters and Michael Herczeg. (2012). Tangicons 3.0: an educational non-competitive collaborative game. In Proceedings of the 11th International Conference on Interaction Design and Children, 144-151.

Hideyuki Suzuki and Hiroshi Kato. (1995). Interaction-Level Support for Collaborative Learning: AlgoBlock—an open programming language. In Proceedings of the first international conference on Computer support for collaborative learning, 349-355.

Hiroshi Ishii, Brygg Ullmer. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. In Proceedings of the ACM SIGCHI Conference on Human factors in computing systems, 234-241.

Hyunjoo Oh, Anisha Deshmane, Feiran Li, Ji Yeon Han, Matt Stewart, Michael Tsai, Xing Xu, and Ian Oakley. (2013). The Digital Dream Lab: Tabletop Puzzle Blocks for Exploring Programmatic Concepts. In Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction, 51-56.

Jeannette M. Wing. (2006). Computational thinking. Communications of the ACM 49, 3: 33-35.

Leslie Eugene Chipman. (2007). Collaborative technology for young children's outdoor education[M]. ProQuest.

Michael S. Horn, R. Jordan Crouser, Marina U. Bers. (2012). Tangible interaction and learning: the case for a hybrid approach. Personal and Ubiquitous Computing 16, 4: 379-389.

Michael S. Horn, Robert J. K. Jacob. (2007). Tangible Programming in the Classroom with Tern. In Proceedings of the CHI '07 Extended Abstracts on Human Factors in Computing Systems.

Neha Katira, Laurie Williams, Eric Wiebe, Carol Miller, Suzanne Balik and Ed Gehringer. （2004）. On understanding Compatibility of Student Pair Programmers. In Proceedings of the SIGCSE technical symposium on Computer science education (SIGCSE’04), 7–11.

Regan L. Mandryk, Kori M. Inkpen, Mark Bilezikjian, Scott R. Klemmer and James A. Landay. (2001). Supporting children’s collaboration across handheld computers. In Proceedings of the ACM SIGCHI Conference on Human factors in computing systems, 255-256.

Robert F. Bales. (1950). A set of categories for the analysis of small group interaction. American Sociological Review, 257–263.

Scott S D, Regan L. Mandryk, Kori M. Inkpen. (2003). Understanding children's collaborative interactions in shared environments[J]. Journal of Computer Assisted Learning,19(2): 220-228.

Timothy S. McNerney. (2004). From turtles to Tangible Programming Bricks: explorations in physical language design. Personal and Ubiquitous Computing, Volume 8, 326-337.

Tomohito Yashiro, Yasushi Harada, and Kazushi Mukaiyama. (2017). Plugramming: A Tangible Programming Tool for Children’s Collaborative Learning. International Conference on Human-Computer Interaction, 398-409.

Acknowledgments

This research is supported by the National Key Research and Development Program under Grant No. 2016YFB0401202, and the National Natural Science Foundation of China under Grant No. 61872363, 61672507, 61272325, 61501463 and 61562063.