TLogic: A Tangible Programming Tool to Help Children Solve Problems

Xiaozhou Deng[1,2], Danli Wang[1(✉)], and Qiao Jin[1,2]

[1] The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China
danli_wang@163.com
[2] School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing, China

Abstract. In this paper, we present TLogic, a tangible programming tool for children aged 6–8. The tool contains two parts: tangible programming blocks and visual tasks with different difficulty levels. Children could use tangible programming blocks to complete visual tasks shown on computer screen. To evaluate our tool, a user study was conducted with 10 children. Results of user study show that the tool could reduce children's cognitive load while solving the tasks and children are more interested in challenging tasks than easy tasks in our tool.

Keywords: Tangible programming · Children · Edutainment

1 Introduction

Programming education has been proved positive for children's development in many aspects, not only in areas such as math and science, but also in language skills, creativity, and social emotional interaction [1]. 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 [2].

However, text-based programming is too difficult for young children due to the rigid syntax and complex programming environment [3]. To reduce the cognitive load of programming, researchers have designed a new interaction method: tangible user interfaces which could make programming easier for young children [4]. With tangible programming tools, children can write programs by assembling the physical objects without keystrokes, which is much easier to involve children in programming [5].

In this paper, we present a new tangible programming tool-TLogic, which contains two visual tasks with different difficulty levels. the reason we provide different visual tasks is that we want to explore children's preference on visual tasks while programming with tangible programming tool.

A. El Rhalibi et al. (Eds.): Edutainment 2018, LNCS 11462, pp. 255–262, 2019.
https://doi.org/10.1007/978-3-030-23712-7_35

2 Related Work

2.1 Tangible Programming Tools

There are many excellent tangible programming tools designed for children, which inspire our work a lot.

Electronic Block [6] uses building blocks to program. It consists of three types of building blocks: sensor blocks, logic blocks and behavior blocks. Each block is embedded with a processor and electronic parts. KIBO [7] is a robotics kit, with which children can construct robots with motors, sensors, and craft materials, and program the robots' actions with some wooden blocks. However, KIBO needs children to capture the programs using a camera manually which is inconvenient for children to use and learn. Tern [8] is a tangible programming language in which children could assemble some puzzle shaped blocks to program a virtual role or a real walking robot. Same as KIBO, Tern also needs children to manually capture the block sequence. Besides, the computer vision technology used in the tool was limited by illumination which might make children feel confused. Also limited by this problem, T-Maze [9] uses a camera to capture the programs that arranged by children and allows children play multi-level maze-escape games by programming. Additionally, real-time feedback is provided on the screen to show the path children have programmed for the characters in game. Strawbies [10] is a real-time tangible programming game designed for children age 5–10. It is an iPad app. Children can play it by constructing physical programs out of wooden tiles in front of an iPad. This work involved children in the design process and developed three different versions to explore different approaches for children easy to use the tool. This design method gave us inspirations.

Inspired by programming tools above, we find tools based on TUI can effectively provide children a programming environment.

2.2 Cognitive Development

Jean Piaget and Bruner's work show us the cognitive ability of children aged 6–8 [11]. According to their work, children aged 6–8 who are in pre-operation stage can't solve problem which contains plenty of intermediate states and complex logic. The theory of cognitive load indicates that real-time feedback which could reduce children's memory load is necessary [12].

The theory of cognitive development helps us make our system effective for children's programming learning and the theory of cognitive load helps us design our tool.

3 Design and Implementation

TLogic is a tangible programming tool with tasks of different complexity. TLogic consists of two parts: visual tasks presented by Unity3D and programming blocks. There would be a detailed rule statement before each task. And the tool provides children with real-time feedback and error massages while programming.

3.1 Visual Tasks

There are two tasks in TLogic. The first only contains sequential structure: two visual characters are on sides of a river. There is a suspension bridge on the river which could be dropped down by a controller. The controller needs to be activated by one of the characters. Children could use programming blocks to let the two characters meet each other. The picture of the first task is shown in Fig. 1.

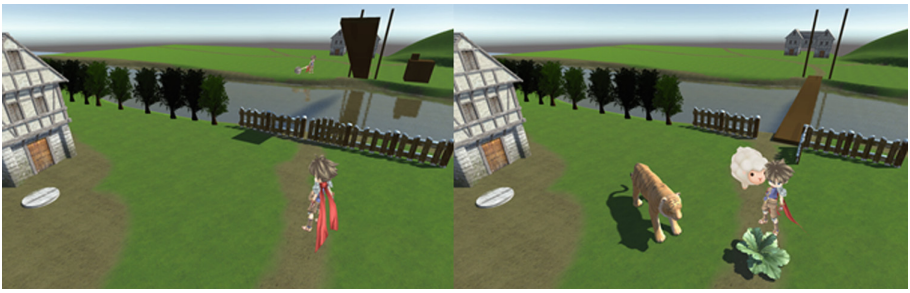

Fig. 1. The two visual tasks.

The second task contains mutually exclusive objects and plenty of intermediate states. A visual character needs carry three objects, a tiger, a sheep and a grass cross a river. The character can only take one object, or nothing cross the bridge. But if the character is on the different side than the tiger and the sheep, there would be an error massage since the tiger would attack the sheep. It is same with the sheep and the grass. The picture of the second task is also shown in Fig. 1.

Considering the cognitive ability of children aged 6–8, we design these two tasks to make a comparison.

3.2 Programming Blocks

Programming blocks are made by 3 cm wooden brick cube with surface texture (Fig. 2). The semantics of blocks are directly corresponding to the operations in visual tasks. For example, in task two, there are four kinds of blocks which are matched with the four kinds way to cross the river in visual game. In our tool we use ReacTIVision framework [13] for real-time recognition of tangible programming blocks.

3.3 Feedback and Error Massage

The design of the feedback and the error massage is important, sometimes decisive to tangible user interface. During completing the second task, children often made some invalid operations while attempting to figure out the solution. For instance, they would try to carry object cross the river even if the object and the character weren't on the same side. It is necessary to make error massages clearly for these invalid operations.

Fig. 2. Programming blocks.

It is hard for children aged 6–8 remember intermediate states. Besides, the presentation of the intermediate states could also encourage children to carry on their programming. We provide children real-time feedback after each of their operation. The intuitionistic feedback reduces children's memory load of the process. Figure 3 show the intermediate state and the error massage.

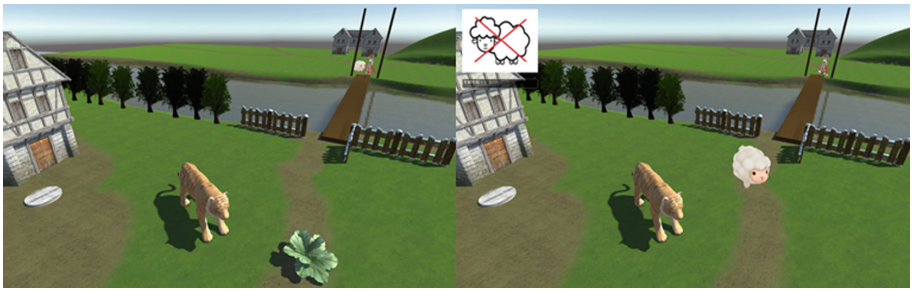

Fig. 3. The intermediate state and the error massage.

4 User Study

To evaluate TLogic, we conducted a lab-based user study with children. In the user study, we were mainly concerned about whether TLogic could reduce children's cognitive load to help them complete more complex tasks and what kind of problems children like.

We have invited 10 children aged 6–8 (7 of them aged 7 and 4 of them are boys) to participate in our study. Experiments were conducted in a spacious room. To capture children's behaviors and voice, a video recorder was set up (Fig. 4).

During the experiment stage, firstly, we showed them how to control visual objects with programming blocks and the feedback on the screen. After these preparations, we let children program to solve the problems. Besides, once a child placed a valid programming block, the tool would log the time stamp.

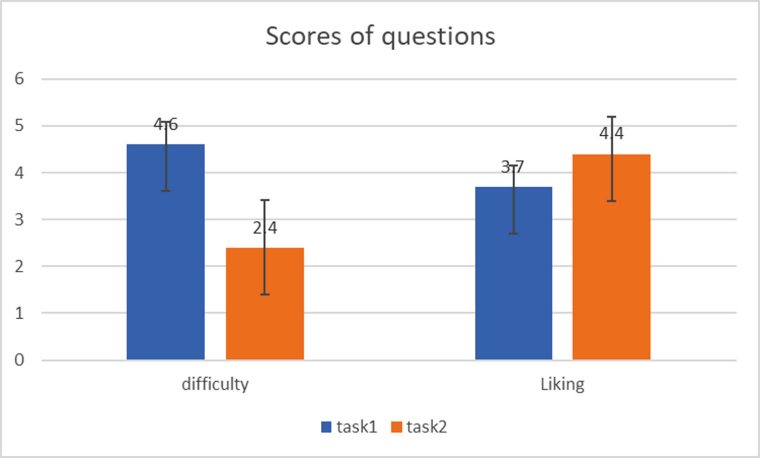

Fig. 4. Scores of Likert-type scale questions. The difficulty in the figure are corresponding to Q1 and Q2. And the liking in the figure are corresponding to Q3 and Q4.

Before programming, we made a short interview to get some basic information of children, such as age, gender and whether got involved with TUIs. After they completed the tasks, we made a brief interview which contains Likert-type scale questions and interview questions. The Likert-type scale is composed to 4 questions scored from one to five, where one means the minimum and five means the max score. The Likert-type scale questions are shown in Table 1.

Table 1. Likert-type scale questions.

Questions	Descriptions
Q1	Do you think the task1 is easy to learn?
Q2	Do you think the task2 is easy to learn?
Q3	How much do you like the task1?
Q4	How much do you like the task2?

The interview questions are shown in Table 2.

Table 2. Interview questions.

Questions	Descriptions
I1	Is there another block sequence of task 1?
I2	Is there another block sequence of task 2?
I3	Which block do you think is the key of task 2?
I4	Do you like this tool and which task do you like more?

5 Results

The results of Likert-type scale questions are shown in Fig. 4.

In Q1, children were asked how task1 is easy for them, and they gave the question average 4.60 (SD = 0.49). Results shown that children thought task1 was easy to them. As for Q2, children gave the question average 2.40 (SD = 1.02). Children thought task2 was not easy for them. To make a comparison, we made a T-Test for Q1 and Q2, and the difference is significant ($p < 0.05$). In Q3, children gave the question average 3.70 (SD = 0.49). Generally, children think task1 was interesting. But in Q4, children gave the question average 4.40 (SD = 0.80). The difference between Q3 and Q4 is significant ($p < 0.05$).

Results of Likert-type scale questions show that children preferred task2 and task2 was more difficult to children.

In order to find whether children did really understand how to solve the two tasks. We asked some interview questions. In I1, all children gave the correct answer. But in interview question I2, only three of them could tell us another block sequence of task2. As for the I3, half of them told us the key is carrying the sheep back. Considering the accidental solutions, we asked children who can't answer I2 and I3 to place the blocks for task2 again. All these kids could place the blocks properly and quickly again.

We have logged all the time stamps of children's valid operations and Fig. 5 shows the average finishing time. The average finishing time of task2 is longer that task1.

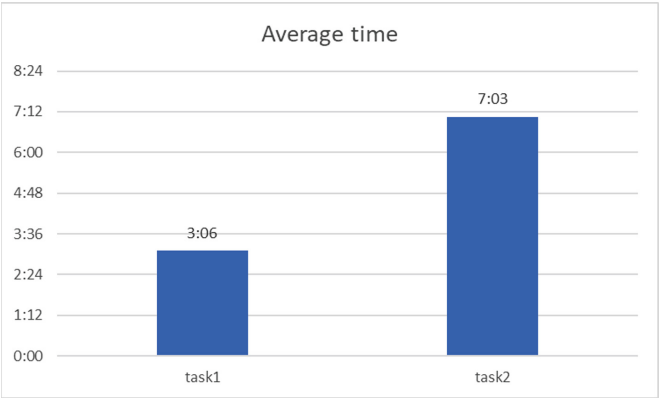

Fig. 5. The average finishing time (minute: second).

Considered all results above, we think TLogic could help children solve and understand the two visual tasks with tangible programming blocks. Besides, children like more difficult task while using TLogic.

6 Conclusions and Future Work

In this paper, we present a tangible programming tool designed for children aged 6–8 – TLogic, which could help children program in a physical form and execute the instructions to solve visual tasks.

According to the results user study, TLogic could reduce children's cognitive load for complex problems and children like challenges with this tool. Besides, after finishing task2 which is more difficult for children, they show self-affirmation and the joy of achievement. Furthermore, we found that children were enthusiasm to logic problems like task2.

To sum up, the design of the content is significant to tangible programming tools. Nowadays, more and more modern technology has provided more and more fantasy interfaces. Indeed, these new user interfaces could gain children's attraction. But we think the visual tasks is important for cognitive development and children may get interested in well-designed visual tasks.

In the future, this work can be improved in several ways. For example, real-time feedback is a significant factor to help children debug the program, so we will provide richer types of feedback. Besides, we need to find more tasks which is appropriate for children's education and apply them into our current work.

Acknowledgment. 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. 61672507, 61272325, 61501463 and 61562063.

References

1. Clements, D.H.: The future of educational computing research: the case of computer programming. Inf. Technol. Child. Educ. Ann. **1999**(1), 147–179 (1999)
2. Wing, J.M.: Computational thinking. Commun. ACM **49**(3), 33–35 (2006)
3. Cockburn, A., Bryant, A.: Leogo: an equal opportunity user interface for programming. J. Vis. Lang. Comput. **8**(5–6), 601–619 (1997)
4. Manches, A., O'Malley, C.: Tangibles for learning: a representational analysis of physical manipulation. Pers. Ubiquit. Comput. **16**(4), 405–419 (2012)
5. Horn, M.S.: Tangible interaction and learning: the case for a hybrid approach. Pers. Ubiquit. Comput. **16**(4), 379–389 (2012)
6. Wyeth, P., Purchase, H.C.: Programming without a computer: a new interface for children under eight. In: Australasian User Interface Conference IEEE (2000)
7. Sullivan, A., Elkin, M., Bers, M.U.: KIBO robot demo: engaging young children in programming and engineering. In: International Conference on Interaction Design & Children. ACM (2015)
8. Horn, M.S., Jacob, R.J.K.: Tangible programming in the classroom with tern. In: Proceedings of the CHI 2007 Extended Abstracts on Human Factors in Computing Systems, pp. 1965–1970 (2007)
9. Wang, D., et al.: E-block: a tangible programming tool for children. In: Adjunct ACM Symposium on User Interface Software & Technology. ACM (2011)

10. Hu, F., et al.: Strawbies: explorations in tangible programming. In: International Conference on Interaction Design & Children. ACM (2015)
11. Bruner, J.S., Lufburrow, R.A.: The Process of Education (1960)
12. Chandler, P., Sweller, J.: Cognitive load theory and the format of instruction. Cogn. Instruct. **8**(4), 293–332 (1991)
13. Kaltenbrunner, M., Bencina, R.: reacTIVision:a computer-vision framework for table-based tangible interaction. In: International Conference on Tangible & Embedded Interaction (2007)