T. Hirashima et al. (Eds.) (2011). Proceedings of the 19th International Conference on Computers in Education. Chiang Mai, Thailand: Asia-Pacific Society for Computers in Education

Robo-Blocks: A Tangible Programming System with Debugging for Children

Nussarin NUSEN, Arnan SIPITAKIAT
Department of Computer Engineering, Chiang Mai University, Thailand
arnans@eng.cmu.ac.th

Abstract: This paper presents a tangible programming system called Robo-Blocks. The system consists of electronic command blocks that can be connected together to form a program that controls the movement of a floor robot. The basis of the design is to make programming accessible to young children. Robo-Blocks adds the ability to debug a tangible program by allowing children to step through each instruction and observe the resulting action. We present case studies that demonstrate how the step-by-step operation can help children to better analyze and debug the robot's action when the outcome is different from their expectations.

Keywords: tangible interface, programming, debugging, robotics

Introduction

Tangible programming is an emerging field that has captured the attention of many researchers. The premise of this area is that tangible interfaces or digital-manipulatives can make programming more accessible to small children. Although there have been a number of on-screen programming environments, such as Logo [2] and Scratch[5], that were designed especially for children, programming in the traditional sense requires the ability to map the on-screen symbolic representation to the actions it produces. This abstraction level can be alienating to young children (and many adults as well). Tangible programming reduces this mental gap by carrying out the programming action through the act of manipulating tangible objects. Tangible programming is, thus, an expression method that taps into the child's existing experience of the physical world. This work presents a tangible programming system called Robo-Blocks. Our main objective is to investigate how to facilitate "debugging": a core activity in programming that engages children in problem solving [2]. Using Robo-Blocks, children with ages of five to eight snap together a series of acrylic command blocks to control a robot that can move around the floor, as shown in Figure 1. There is no need to use a computer in this process.

1. Tangible Programming

There are two main threads of tangible programming environments: implicit and explicit. Implicit programming embeds the programming action within the target object. Topobo [4] embraces a "record and play" paradigm by creating "kinetic memory" to each piece of a specially designed robotic construction kit. Topobo allows children to create kinetic structures by manipulating the physical structure itself.

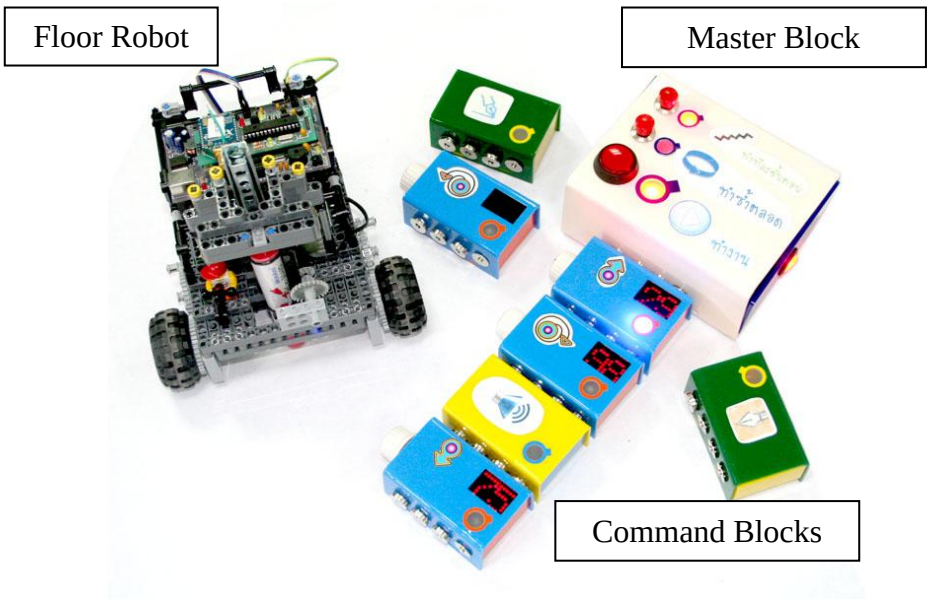

Figure 1: Robo-Blocks consists of command blocks connected to a master block which interprets the program and wirelessly sends the commands to the floor robot.

This research focuses more on explicit tangible programming, which offers a separation between the programming logic and the programmed object. There is a set of tangible objects or blocks that represents primitives of a computer program, which then get translated into actions controlling a separate physical object. The Button Box [3] developed by Radia Perlman at the MIT Logo Lab in the mid-1970s is most likely the first example of this paradigm. The system was design to control a "floor turtle", a robot that moves around with a pen attached. Perlman later developed the Slot Machine [3], which was conceptually a significant step forward from the Button Box. Instead of buttons, plastic cards were used to represent the possible turtle actions. Children can place many cards onto racks, thus creating a command sequence. The cards used with the Slot Machine provided a much more concrete way for children to think about and, more importantly, alter their program. In essence, the Slot Machine gave a way for children to become involved with "debugging": one of the most important cognitive activities in learning through programming.

More recent research projects that follow this path includes Tern [1], which allows children to program a moving robot by putting together wooden blocks each representing a robot command. A camera then interprets the block sequence and tells the robot to execute the command sequence.

However, despite the increase research interest on tangible programming, we were surprised to discover that very little attention has been given on further improving the debugging process. In fact, the Slot Machine from 1976 remains the best example of how debugging can be implemented!

2. Programming and Debugging with Robo-Blocks

The Robo Blocks system consists of acrylic boxes embedded with a custom designed micro-controller circuit. Each block represents a command that can be used to control the floor robot. They each have a set of magnetic connectors both on the top and bottom sides allowing children to easily snap the blocks together creating a command chain.

Block Name	Description
Movement Blocks	
Forward	Moves the robot forward
Backward	Moves the robot backward (without changing its heading)
Left turn	Turns the robot to the left
Right turn	Turns the robot to the right
Pen commands	
Pen down	Puts the pen down causing the robot to leave a trail when it moves
Pen up	Retracts the pen, preventing it from leaving any trails
Auxiliary commands	
Beep	Tells the robot to beep

Table 1: A list of command blocks used in this work

The movement blocks feature a built-in screen showing a number ranging from 0 to 99. Children can turn a knob on the side of the block to change this value. The value determines the amount of time a movement block performs the action.

The command blocks must be attached to a master block. This master block automatically discovers the number of blocks, the type of each block, and the sequence of the blocks. The master block then sends the commands to the floor robot over a wireless communication link.

The floor robot was built using Lego Bricks, except for the pen levitation control mechanism where a custom designed mechanical structure was used. The robot's movement is controlled using a GoGo Board, an open robotics platform for children [6]. The GoGo Board was selected because of its simplicity and it supports wireless communication.

Robo-Blocks allows children to debug their program through the step-by-step button on the master block. When something goes wrong or happens unexpectedly, children can choose to analyze the situation by executing one command at a time.

3. Evaluation

Twelve students with ages between five and eight years old participated in the evaluation process. There were five males and seven females. None of these students had any prior programming background. Six students worked in groups of two. Thus, a total of nine cases were performed. Each case lasted approximately two hours.

There were two activities designed for this experiment. (A) Solving a Maze. After a brief introduction of Robo-Blocks, students were challenged to put together a program (sequence of blocks) that would guide the robot through a given path. The condition was that the robot must reach the destination without touching the rim of the path. A straight line was used as an introductory path followed by an "L" path and a slightly more difficult "U" shaped path. (B) Turtle Geometry. This is the same activity introduced by earlier works such as the Button Box and the Curlybot described earlier where students program a robot to draw geometrical shapes on the floor. Students learn to use the "pen-down" and "pen-up" command blocks that determine whether or not the robot leaves a trail as it moves. The researcher then challenges the student to draw simple shapes such as a triangle and a square.

Data used in the analysis of this research were collected from three main sources. First, all the cases were recorded on video. The footage was extensively used to analyze the common themes among participants. All students were interviewed with questions that indicated how well they understood what they were doing and how they felt about the tools they were using. Lastly, the researchers kept a journal of each case study that reflected the important events that were observed.

4. Findings

4.1 Usability and Engagement

All the participating students were clearly attracted by the design and use of Robo-Blocks. Few students were slightly intimidated initially by the robot. The hesitation disappeared after understanding what Robo-Blocks can do. The affect of pressing the run button and seeing the robot move was clear (see figure 2). Given that none of the students had prior exposure to robotics, this new experience is both exciting and engaging.

Figure 2: Images of students interacting with the Robo-Blocks system

Students were able to understand the function of the dial located on the action blocks. For example, every student was comfortable with the concept of changing the value on a "MOVE-FORWARED" block to determine how far the robot should move. Students understood that if they want the robot to move forward and then return back to the starting point they need to make sure the numbers on the "MOVE-FORWARED" and the "MOVE-BACKWARDS" blocks are equal. Thus, the concept of a "command parameter" was simple and comprehensible.

4.2 Step-by-Step and Debugging

When a program executes, the student usually pays attention to the robot more than the blocks. The robot action is fast and students can easily lose track of what block the robot is running. This is when the step-by-step ability became useful. Each time the student pressed the "Step" button, one block was executed. This controllable behavior helped students to better synchronize their thinking with the actual program being executed.

We have also found that the ability to step through instructions can help with debugging. This observation was clearly seen when two students were trying to draw a triangle. The desired action was for the robot to "MOVE FORWARD", "TURN LEFT", "MOVE FORWARD", "TURN LEFT", and "MOVE FORWARD" (See figure 4). But they incorrectly used a "TURN RIGHT" block on the second turn. Thus, the result was a zigzag shape instead of a triangle. At that time, once the robot started moving, the students could not pin point which block was causing the problem. But the cause became apparent once they stepped through the code one block at a time.

We have observed the use of step-by-step as means for slowing down the execution and/or debugging an error in six out of the nine cases we have conducted. Students in the remaining three cases who did not use the stepping feature preferred to rely on their observations and the effect of replacing blocks to proceed with their work. This later group accomplished less work than the former. Among the three groups that did not use the step-by-step function,

two were not able to finish drawing the shapes in activity (B). On the other hand, only one of the six cases that used step-by-step did not finish the activity.

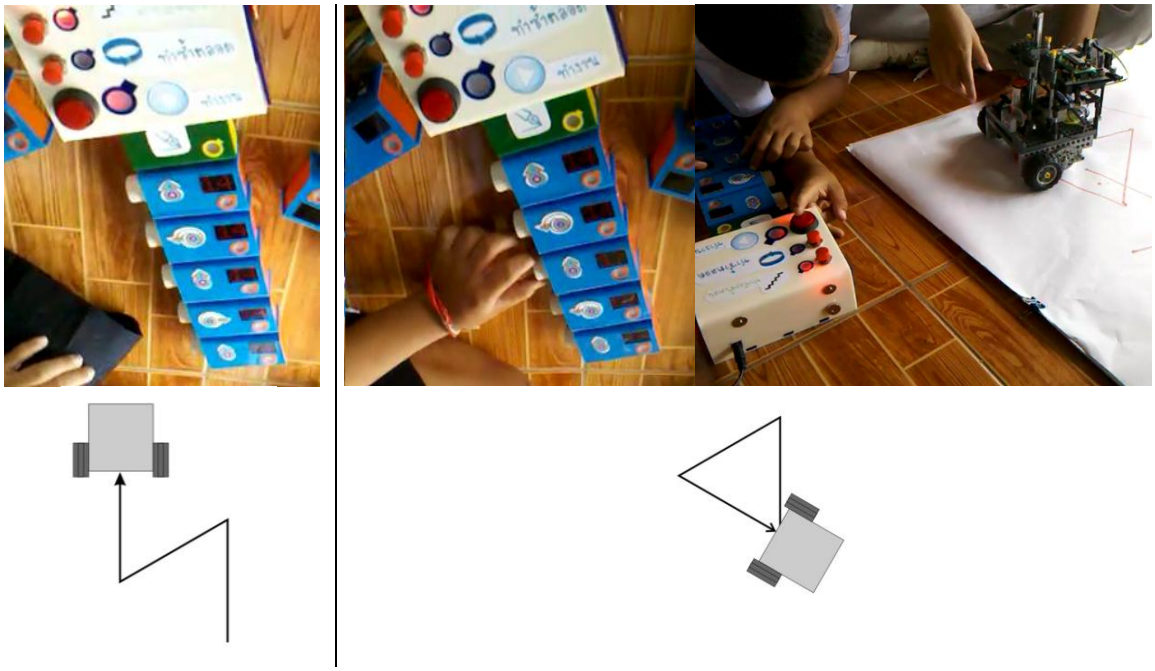

Figure 4: The left column shows the incorrect triangle program and the zigzag result. The right colum shows the corrected program, which was assisted by utilizing the step-by-step feature.

5. Conclusions

This work has contributed to the field of tangible programming by demonstrating how debugging can be designed and implemented. Robo-Blocks is a tangible programming framework that offers a step-by-step feature to run one instruction at a time. We have showed how students appreciate this ability and how it can play an important role in the learning process. Thus, we believe debugging abilities should be considered seriously when designing any tangible system for children. Other debugging concepts such as break-points and variable watchers could be explored with more sophisticated programs.

References

[1] Horn, M., Solovey, E. T., Jacob, R.J.K. (2008). Tangible Programming and Informal Science Learning: Making TUIs Work for Museums, *In Proc. IDC 2008 Conference on Interaction Design for Children*.

[2] Papert, S. (1980/1993). *Mindstorms: Children, computers, and powerful ideas* (1st and 2nd ed.). Cambridge, MA: Basic Books.

[3] Perlman R (1976) Using computer technology to provide a creative learning environment for preschool children. *Logo memo no 24, MIT Artificial Intelligence Laboratory Publications 260*, Cambridge, Massachusetts, USA.

[4] Raffle, H. S., Parkes, A. J., & Ishii, H. (2004). Topobo: a constructive assembly system with kinetic memory. *In Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 647–654). Vienna, Austria: ACM.

[5] Resnick, M. , Maloney ,J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, M., Rosenbaum, E., Silver, J., Silverman, B., Kafai,Y. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11), 60–67.

[6] Sipitakiat, A., Blikstein, P., & Cavallo, D. P. (2004). GoGo board: augmenting programmable bricks for economically challenged audiences. *In : ICLS '04, Proceedings of the 6th international conference on Learning sciences* (pp. 481-488). International Society of the Learning Sciences.