RoboTable2: A Novel Programming Environment using Physical Robots on a Tabletop Platform

Masanori Sugimoto Tomoki Fujita Haipeng Mi
Interaction Technology Laboratory
Department of Electrical Engineering & Information Systems
University of Tokyo
7-3-1 Hongo Bunkyo-ku, Tokyo, 113-8656, Japan
{sugi, fujita, mi}@itl.t.u-tokyo.ac.jp

Aleksander Krzywinski
University of Bergen
Fosswinckelsgate 6
5020 Bergen Norway
aleksander.krzywinski@
infomedia.uib.no

ABSTRACT
In this paper, we propose a novel environment called RoboTable2 that supports users with limited programming knowledge or experience in conducting robot programming. We have devised a tabletop platform that can simultaneously recognize multi-touch input and track physical robots, enabling users to conduct robot-programming tasks in an intuitive manner. A user study comparing the proposed and a conventional graphical programming environment was conducted, involving ten university students with no programming experience. The effects of the proposed environment were clarified via video analysis, questionnaires and usage logs. A pilot study was also conducted to verify how easily users could design and develop applications on RoboTable2. The lessons learned with respect to design guidelines for the proposed programming environment and issues for investigation are discussed.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation** (e.g., HCI)]: User Interfaces - Input devices and strategies, Interaction styles, Prototyping.

General Terms
Design, Human Factors

Keywords
Programming, technique, multi-touch tabletop environment, tangible robot.

1. INTRODUCTION
Due to the recent development of ICT (Information and Communication Technologies), many kinds of digital learning materials have been introduced in schools. An example of such materials is simulation environments that allow learners to learn about complicated problems such as environmental problems or urban planning. The merits of such environments have been discussed in several literature (e.g. [1]) reporting that learners can explore solutions through try-and-errors processes, which deepens their knowledge and understanding about the problems. Moreover, it is also reported that introducing robots or tangible objects can raise learners' motivations and engagement for their learning [2].
We have so far been investigating physical media using robots and tangible objects that allow learners to design and develop learning contents on simulation environments by themselves, which enhances their self-directed learning [3]. To design such physical media, it is necessary that learners can easily conduct robot programming for simulations by themselves. Supporting a programming task is not easy because it includes several issues such as user interface issues (easy to use), learners' comprehension of logical concepts (e.g. conditional branch, repeat) and so on. As learners do not always have sufficient experiences on using computers or knowledge about programming, it is inevitable to provide such learners with a programming environment where they can easily create a program that controls behaviors of a robot.

In this paper, we propose a novel environment for intuitive robot programming on a tabletop platform. The proposed programming environment called RoboTable2 is implemented by extending our existing tabletop platform called RoboTable [4] which can accept multiple finger input and conduct robot tracking. A user can create a program in an easy manner, for example, he/she can define their intended behaviors of a robot by directly grasping and moving it on a tabletop surface. It is also possible for a user to easily create condition statements (e.g. "if a robot collides against a virtual wall") or other control structure statements by manipulating a virtual item displayed on the table using finger gestures. Therefore, RoboTable2 is expected to lower barriers of programming for children or university students who have no or less programming experiences.

Many programming techniques or support systems for novice programmers or children have been proposed. Examples of them are graphical or tangible programming environments. One difference between these existing studies and the proposed environment is that it allows users to program robots in an intuitive manner by grasping and moving them and conducting figure gestures on a tabletop platform. Another difference is that the proposed environment allows a user to manipulate both

Full presentation, *ACE'2011* - Lisbon, Portugal

physical and virtual items simultaneously. As far as we know, there is no such robot programming environment so far.

We have conducted comparative user studies between the proposed and graphical programming environments. University students with no programming experience participated in two experiments: one using a virtual robot and the other using a physical robot. The evaluations were based on post-experimental questionnaires and interviews and video analyses recorded during the user studies. Significant differences were found between the two environments in relation to some of the measurement items and the number of programs (in this paper, we define a program as a set of program blocks, as explained Section 3.2). Through an initial proof-of-concept study, it was confirmed that users could easily design and develop applications using RoboTable2. We carefully investigated the experimental results and clarified problems to be solved, issues to be investigated and lessons learned toward design guidelines of this new programming environment.

The organization of the papers is as follows. In the next section, related works are described. In Section 3, the proposed programming technique is explained. Section 4 shows the comparative studies, the results and discussions based on them. Section 5 describes a proof-of-concept pilot study. Section 6 gives conclusions of this paper.

2. RELATED WORK

A lot of programming environments for people with less programming experiences have been proposed so far. In this section, several previous studies related to the proposed environment, mainly tangible programming, are described.

There are many projects that use physical items to be connected or stacked for constructing programs. AlgoBlock [6] allows children to build a program by connecting electronic building blocks to form a sequence of program steps. The output of AlgoBlock programs is displayed on a computer screen. Quetzal [7] is designed for young elementary school children (1st or 2nd graders) and helps their programming by connecting physical plastic tiles to control a LEGO Mindstorms RCX brick. A constructed program by children is captured with a digital camera and complied into machine readable codes using image processing techniques. Tern [8] inherits features of Quetzal and is used for controlling simple virtual robots on a computer screen. Tangicons [9] and Digital Construction Kits [10] allow children to create programs by stacking cubes. Tangicons follows ideas proposed in Quetzal [7] and visual representations (sound, light etc.) are printed on faces of cubes so that younger children can easily understand their meanings. Digital Construction Kits is designed based on functional programming concepts and supports children in exploring the logic of everyday electronic devices through hands-on learning. Teh et al propose a system that allows children to program robots using the sense of touch [11]. Electronic Blocks [12] includes three types of blocks (sensor blocks, action blocks and logic blocks). By stacking different blocks, children can create various physical structures with different behaviors. roBlocks [13] uses modular building block units including sensor blocks and actuator blocks, and allows users to construct a physical robot. By changing input values to actuator blocks obtained by sensor blocks based on the hop counts between them, it becomes possible to flexibly change robot behaviors. In GameBlocks [14], a user is asked to choose tangible instruction blocks and arrange them on trays. Then, the user can control the movement of a robot by the instructions of individual blocks in the arranged order.

curlybot [15] is a graspable robot that can accept physical input. When a child holds and moves curlybot, it memorizes the trajectory and plays it back in a physical space. Topobo [16] is a 3D constructive assembly system with kinetic memory that allows users to record and replay physical motions. Although curlybot and Topobo allow a user to conduct programming tasks with physical input like grasp and move actions, he/she cannot create a complicated program such as conditional branches. A physical programming tool [17] enables young children to create their own interactive environment in an intuitive manner. The tool is not for robot programming but for controlling lights or sounds for their storytelling. In Pleto [18], when a child touches a part of a body of a robot, its possible actions are displayed on a computer screen based on "thought bubble" metaphors. He/she can conduct programming by choosing one of the actions and linking it to the touched part. TurTan [19] is a tabletop programming environment. By arranging tangible blocks on a table, children can control moves of a virtual turtle displayed on it. Turtan is not used for programming a physical robot.

There are many works related to robot control techniques using a multi-touch tabletop platform. Examples of them are to use finger gestures for creating vector fields followed by robots [20], a controller for home appliances including mobile cleaning robots [21], and design guidelines of natural gestures [22]. Puppet Master [23] uses a physical puck for designing character motions. When a user translates and rotates a tangible puck, motions of a virtual character are simultaneously generated and visualized on a tabletop surface. Guo et al propose techniques for controlling a remote group of robots with finger gestures and tangible input devices (stuffed toy) [24]. In MuMoMuRo [25], a similar idea to [24] is described. In Tangible Bots [26], interaction techniques with motorized robots are described. These predecessors do not support users in their robot programming tasks including conditional branches.

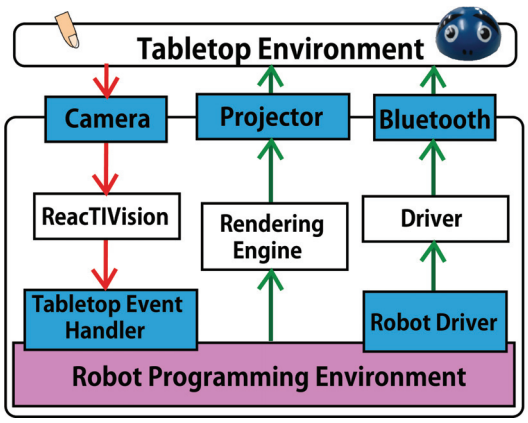
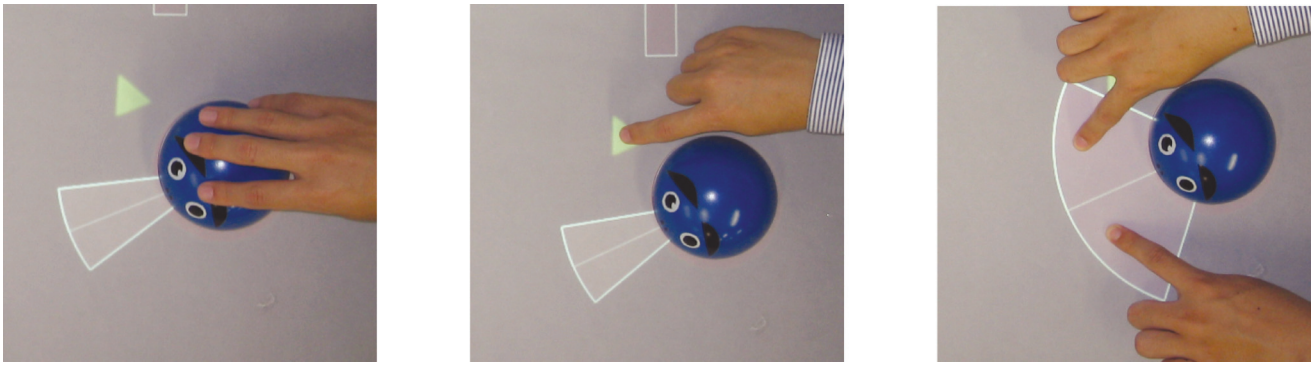


Figure 1: System configuration of RoboTable2.

3. PROPOSED PROGRAMMING ENVIRONMENT

3.1 System Overview

RoboTable2 has been constructed by extending our tabletop platform called RoboTable [4]. RoboTable2 integrates DI (Diffused Illumination) [27] and FTIR (Frustrated Total Internal Reflection) [28] so that it can simultaneously recognize multi



(a) (b) (c)

Figure 2: Input techniques on RoboTable2. (a) "grasp and move" (b) "touch" and (c) "resize and reshape".

finger touches and conduct robot tracking. For tracking physical robots, an image recognition library called reacTIVision [29] that can identify a fuducial marker attached to the bottom of each robot is used. A robot is controlled by RoboTable2 via bluetooth communications. The configuration of the RoboTable2 environment is shown in Figure 1.

3.2 Event Driven Programming Model

In designing RoboTable2, we use an event driven programming model. In this model, a user is requested to define an event and its corresponding behavior of a robot when they create a program. A pair of the event and the behavior is called a program block composing a part of a program. The created program block is registered in the database of RoboTable2 (Figure 5). When a program is executed, RoboTable2 conducts matching tasks between the current situation that a robot encounters and events of all the registered program blocks. When RoboTable2 finds a program block corresponding to the current situation, it sends a control command to the robot to activate the behavior of the block. For example, when a robot detects a virtual obstacle and RoboTable2 finds the corresponding program block that includes a behavior "turn right at 90 degrees", RoboTable2 sends the control commands to the robot so that it shows the behavior.

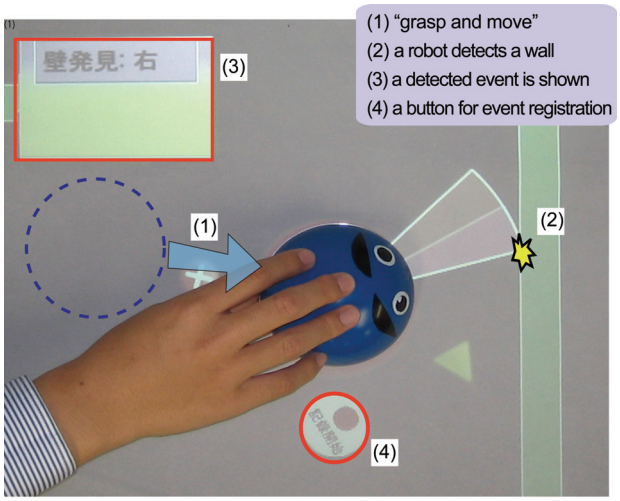


Figure 3: Registration of a recognition event. (1) A user first grasps and moves a robot so that (2) it detects a wall. When a user finds that (3) the recognition event displayed as "if a robot detects a wall at the right direction" on the table is what he/she intends, (4) he/she presses a button shown around the robot for event registration.

3.3 Programming Technique

A user can create a program on RoboTable2 by "grasp and move" actions or finger gestures as shown in Figure 2. He/she first defines an event, then a behavior of a robot, and finally creates a program block. A user repeatedly creates a program block so that a robot is controlled as he/she intends.

3.3.1 Recognition Event

A recognition event is an event meaning that a robot detects or collides with an object. In order to define a recognition event, a user first sets a sensing area within which a robot can detect a physical or virtual object. As shown in Figure 2 (c), a user can change a radius and an angle of a sector-shaped area visualized around a robot by using fingers. If a user makes the radius longer and the angle wider, he/she can increase the robot's capability to detect an object.

Figure 3 shows how a user registers a recognition event. Suppose that a user wants to change a behavior of a robot when it detects a virtual wall at the right direction. The user first grasps the robot and moves it close to the wall, or drag the wall to its sensing area. When the wall touches the sensing area of the robot, RoboTable2 identifies that the robot detects the wall. Then RoboTable2 automatically displays a window explaining the recognition event as "a robot finds a wall at its right direction" on the table (Figure 3, upper left). A user touches a registration button to register the recognition event or move the robot (or drag the wall) in a different way again to change the event.

A user can create a recognition event that includes multiple objects, for example "the robot finds a wall in front of it and another robot at its left direction". However, to reduce the complexity of situations for a novice user, in the current implementation, one object detected first by the robot is used for a recognition event. A user can also define an empty recognition event meaning that no object is detected by a robot. This event is repeatedly invoked by RoboTable2 during program execution and overridden by the other recognition events when one of them happens.

3.3.2 Behavior

After registering a recognition event, RoboTable2 prompts a user to define a behavior of a robot corresponding to the event. Figure 4 shows how a user creates robot's behaviors for a new program block. A user grasps a robot and creates its behavior by moving it on the table. RoboTable2 simultaneously tracks the position and rotation of the robot being moved and converts the behavior into a sequence of control commands composed of "move forward", "move backward" "rotate clockwise" and "rotate counterclockwise". In Figure 4, a user moves a robot forward to define a behavior corresponding to the recognition event ("if a robot finds a wall at its right direction"). Then RoboTable2 adds a description of the behavior to the window (Figure 4, upper left). A user confirms whether he/she could create his/her intended behavior. If it is acceptable to the user, he/she touches a behavior registration button around the robot. Otherwise, the user can redefine behaviors of the robot by moving it again. When a user touches the button, he/she can see a message explaining that a new program block including the recognition event and the behavior has been created and registered. He/she can create multiple program blocks by repeating the same procedure.

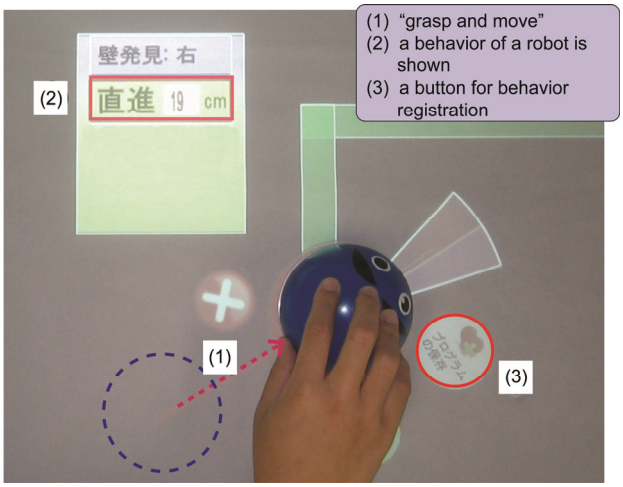


Figure 4: Registration of robot's behaviors. (1) A user first grasps and moves a robot. (2) Then, a behavior given by him/her is displayed on the table. (3) If the displayed behavior is what the user intends, he/she presses the button for registration.

3.3.3 Editing Program Block

When a user wants to revise a program block, he touches a program block edit button shown around a robot. Then all the program blocks created by the user appear on the table. The user can select or drag one of the blocks with his/her finger. The user can revise a selected program block by following the same manipulations described in the previous sections. If the user wants to remove a selected block, he/she drags it to a trash icon shown at the corner of the table.

3.3.4 Program Execution

A user can execute a program by touching an execution button shown around a robot. During the execution, RoboTable2 always checks if there exists an object in a sensing area of a robot. If RoboTable2 finds an object in the area, it conducts verification between the identified event and recognition events of the program blocks created and registered by a user. When RoboTable2 finds the same recognition event, it controls a robot based on the behavior corresponding to the recognition event in the program block. Otherwise RoboTable2 uses the behavior of the empty recognition event. To stop the execution, a user touches a stop button shown around a robot.

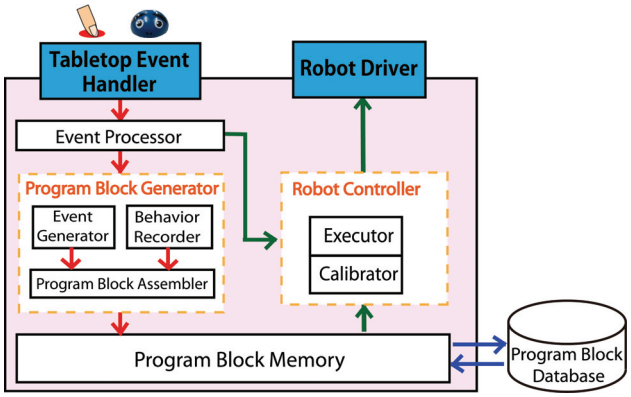


Figure 5: Software architecture of RoboTable2.

3.4 Software Architecture

The software architecture of RoboTable2 is shown in Figure 5. The system obtains positions of a robot and user's fingers through Tabletop Event Handler and controls the robot via RobotDriver. An event from Tabletop Event Handler is analyzed by Event Processor to check if it is a recognition event, finger gesture event (button press or drag by a user), or tangible object tracking event (robots or the other physical items). Based on types of events identified, the subsequent procedures are conducted by Program Block Generator and Robot Controllers in the following ways.

- *Generation of program blocks*:

When Event Processor identifies that a new event from Tabletop Event Handler is a recognition event, it invokes Program Block Generator and shows a recognition event registration button. If a user touches this button, Event Generator is invoked and the event is registered as a recognition event. Then Behavior Recorder is invoked and prompts the user to define a robot behavior for the recognition event. He/she grasps the robot and moves it on the table as he/she likes. Behavior Recorder simultaneously converts the movement into control command sequences which are appropriate for the robot drive mechanism (two-wheel or four-wheel). If a user touches a behavior registration button, Program Block Assembler is invoked to create a new program block.

- *Registration of program blocks:*

Program Block Assembler assembles the recognition event and the behavior into a program block and shows a program block registration button on the table. When a user touches the button, it is registered into RoboTable2 Program Block Database via Program Block Memory.

- *Execution of program blocks:*

When Event Processor identifies that a new event from Tabletop Event Handler is a user's touch on a program execution button, Robot Controller is invoked. It accesses to Program Block Memory and loads the program blocks registered on it. When Event Processor finds a recognition event during program execution, it sends the event to Robot Controller. Robot Controller then conducts verification between the received recognition event and that in the program blocks. When RobotController finds the corresponding event in the program

blocks, Executor executes the matched program block by sending the control command sequence to the robot. During the execution, the robot's positions are tracked. When RoboTable2 identifies errors between registered behaviors and the current robot trajectory, Calibrator revises the control command sequence to reduce the errors and adjust to the behaviors.

4. EXPERIMENTS AND EVALUATIONS

4.1 Overview

The evaluation of Robotable2 was conducted. For comparative user studies, a graphical programming environment was developed so that subjects could manipulate and assemble graphical items with their fingers. Using different tangible programming methods such as stacking or connecting physical blocks was first considered for the user studies. However, as this was the first user studies and we wanted to know effects of tangible input on a multi-touch tabletop environment, we implemented a widely used graphical programming environment for the comparisons.

The remarkable difference between the proposed RoboTable2 and graphical programming environments was that the former allowed users to define behaviors of robots by directly grasping and moving it and the latter asked users to input parameter values by touching sliders, buttons or menus displayed on the table as shown in Figure 6. Subjects were asked to conduct a robot programming task on given mazes: they had to create a program that controlled a robot by avoiding a virtual wall to reach the goal (Figure 7).

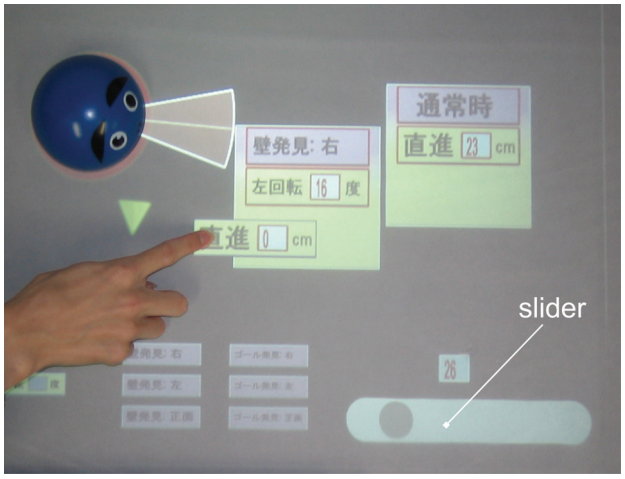


Figure 6: A graphical programming environment used for comparative user studies. A subject uses sliders, buttons or menus for setting and selecting parameter values.

Four different mazes whose difficulty levels were confirmed to be almost the same by experimenters were used. The start and goal points were the same in these mazes. Programs that the subjects created were executed at the start point and terminated at the goal point.

10 university students (male: 5, female 5, average age 20.3) who had no programming experience were recruited from different departments (literature, liberal arts, law, and education) of our university. We first explained how they could create programs by using individual programming environments. Then they tried these environments. When we confirmed that each subject got accustomed to them after his/her 10 or 15 minutes use, we started experiments for each subject.

In each experiment for each subject, one of the four mazes was selected randomly to exclude his/her learning effect. A programming environment (proposed or graphical) and a robot (physical or virtual) were also chosen randomly for avoiding order effects. We measured how long each subject spent for completing a program and conducted video recording during the experiments. Post-experimental questionnaire surveys were carried out using the following four measurement items:

1. Easiness level of understanding programming methods ("Understandability")
2. Easiness level of defining recognition events ("Recognition event")
3. Easiness level of defining robot's behaviors ("Behavior")
4. Satisfactory level of the performance of created programs ("Satisfactory level")

The item 4 ("Satisfactory level") asked subjects if they created programs that could control a robot as they intended. In the questionnaire survey, a seven-level likert scale was used (1: strongly disagree, 2: disagree, 3: weakly disagree, 4: neutral, 5: weakly agree, 6: agree, 7: strongly agree). The survey also asked subjects to describe about problems, suggestions or whatever else that they found during the experiments.

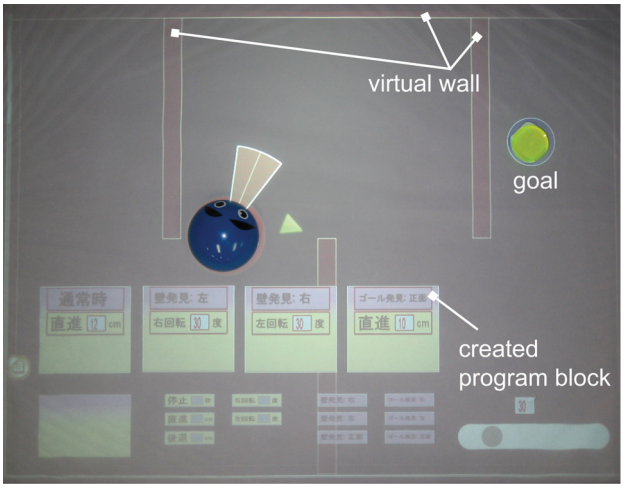


Figure 7: An example a maze used in the experiments.

4.2 Results of Experiment

In Experiment1 that used a virtual robot, the Wilcoxon signed rank test, which was one of the non-parametric statistical hypothesis tests for ordinal scale data, was applied to the scores of the questionnaire assessed by the subjects on the proposed and graphical programming environments. The results are shown in Figure 8. The test clarified that significant differences were found between the proposed and graphical programming environments as for the "Recognition event" (p <.01), "Behavior" (p <.05) and "Satisfactory level" (p<.01).

Figure 9 and 10 show elapsed times for programming and numbers of program blocks created in the proposed and graphical environments, respectively. By using the Smirnov test and F-test, we first confirmed that the data obtained in these two environments did not support hypotheses that they were not normal distributions and did not have an equal variance. A paired t-test revealed that a significant difference was found between the

proposed and graphical programming environments for the numbers of program blocks (p <.05), but no significant difference was found for their elapsed times.

4.3 Discussions

Through the experiments, it was confirmed that RoboTable2 allowed the subjects to more easily define recognition events, define robot's behaviors and gain higher satisfactory levels of the performance of created programs than the graphical programming method, although no significant difference was found as for the understandability of the programming methods. Significant differences were found as for the number of program blocks created by each subject. These results indicated that RoboTable2 allowed users to express recognition events for conditional statements and behaviors of robots in more intuitive and easier ways than the graphical programming method. In this section, problems to be solved, issues to be investigated and lessons learned from the experiments are discussed.

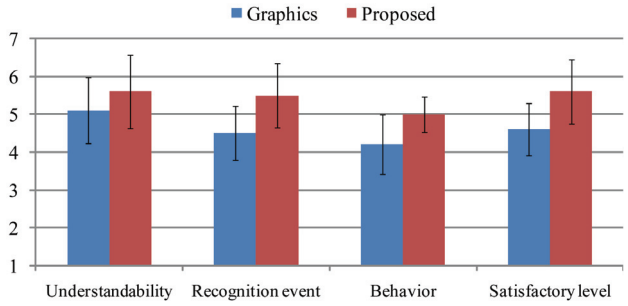


Figure 8: Questionnaire results.

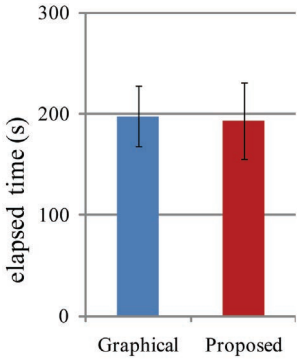


Figure 9: Time spent for programming by each subject (unit: second).

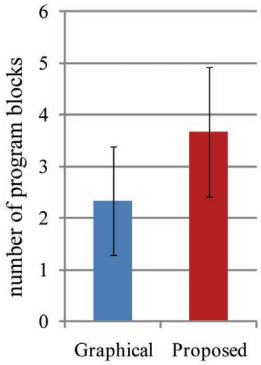


Figure 10: Number of program blocks created by each subject .

· Information display

When a subject defined a behavior of a robot using the proposed environment, it displayed command sequences including numerical parameters (e.g. rotate left at "90 degree", move forward "10 cm") on the table as shown in Figure 4. If a user wanted to set more exact values for control command parameters (for example, not "10 cm" but "11 cm" for move forward, or not "90 degrees" but "85 degrees" for "rotate), he/she had to grasp and move the robot repeatedly for the adjustment. Two subjects told that it was a frustrating task. They had to repeatedly and interchangeably look at the robot and displayed parameter values, which made them irritating. We found through video analyses that subjects of the proposed environment (i.e. RoboTable2) spent a considerable time for this kind of adjustment task. In order to reduce subjects' burden in setting exact parameter values, more efficient and less frustrated ways for displaying numerical parameter values must be examined.

· Toward design guidelines of the proposed programming environment

The experiments indicated that the graphical programming environment was suitable for setting exact numerical parameter values for control commands such as a moving distance or rotation angle by using a slider. On the other hand, the proposed environment seemed useful for defining less exact but qualitative behaviors of a robot in an intuitive manner. Thus we have to investigate how these two environments should be integrated so that users can conduct programming tasks in more efficient ways.

The proposed environment could accept physical and virtual input for creating a program: a subject could not only grasp and move a physical robot, but also resize and reshape its sensing area with fingers. We think that accepting physical and virtual input simultaneously could fully utilize advantageous points of the proposed environment that had tangible robots on a multi-touch tabletop platform. However, many existing studies related to tangible or tabletop programming restrict to either physical input (e.g. topobo, curlybot) or virtual input (e.g. Turtan).

When a system accepts physical and virtual input for robot programming, it may be possible for a user to create a program more intuitively and efficiently, and easily control a robot as he/she intends. On the other hand, a user has to consider both physical and virtual input simultaneously, which may make a programming task more complicated and difficult. Therefore, when we introduce a new programming environment that integratively uses physical and virtual input like the proposed environment, we need a design guideline for it. Actually, we have just started initial user studies focusing on this purpose.

5. PILOT STUDIES: APPLICATION DEVELOPMENT BY USERS

A pilot study was conducted to confirm how easily users could design and develop applications on RoboTable2 by themselves. It was the first feasibility study and its aim was not to clarify effects for supporting users with less programming experiences under controlled experimental conditions, but to obtain feedback and comments for examining possibilities of RoboTable2 as an application development environment by a user him/herself. Thus, when we asked subjects to participate in the study, their programming experiences were not considered, which was different from the experiments in Section 4. Five university students were given two robots and asked to construct a traffic simulator. Before the study, the subjects were instructed about the usage of RoboTable2 and tried it for about 10 minutes.

For developing a traffic simulator, the users first drew roads with their fingers, and selected a virtual item such as a traffic signal and a road sign from a pallet and placed them at any location on the table. They then defined behaviors of each robot. If they wanted to demonstrate a careful automobile driver, the users made a sensing areas of a robot larger so that the driver could identify stop signs at a distant location from them, and vice versa. By assigning different lighting times to red, yellow and blue lights of each traffic signal, users controlled traffic flows of their simulators.

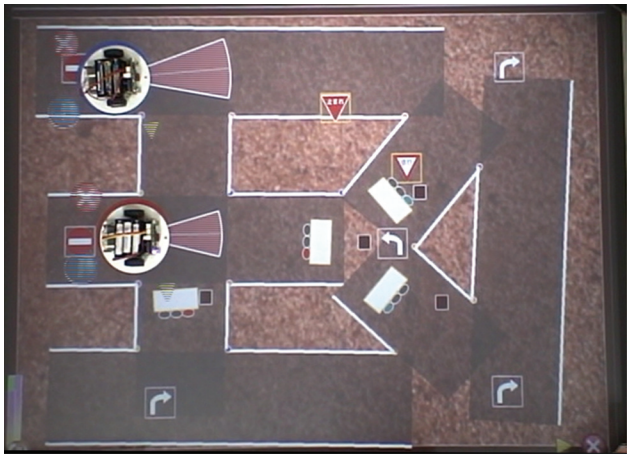


Figure 11: An example of a traffic simulator developed by one subject.

The average elapsed time for constructing traffic simulators by five subjects was 16 minutes 43 seconds (maximum: 24 minutes 11 seconds, minimum: 10 minutes 36 seconds). The average numbers of virtual items used and program blocks created were 6.8 (maximum: 11, minimum 4) and 16.8 (maximum: 24, minimum: 11), respectively.

Figure 11 shows an example of a traffic simulator by one of the subjects. He tried to design and develop it for demonstrating traffic situations in a downtown area including complicated street crossings. He spent for 10 minutes 36 seconds, used 9 virtual items and created 20 program blocks to complete the simulator.

Positive comments from the subjects were "it was not difficult for conducting robot programming on RoboTable2", "it sounds easy to develop a simulator with a moderate level of complexity", "it was good to execute a simulator in different settings by easily changing its simulation parameters (e.g. positions of road signs, numbers of traffic signals) with finger gestures", "using physical robots increased the reality level of simulations" and "if we use more robots, we can create more realistic traffic simulations". These comments indicate that RoboTable2 has possibilities to allow users by themselves to design and develop applications using physical robots in an easy and intuitive manner.

Major negative comments from the subjects were "it was desirable to use a small-sized robot or a larger table for constructing more complicated traffic simulations" and "higher degrees of freedom must be given for setting a sensing area of a robot and defining its recognition events and behaviors". The latter comment is related to the performance of the robot. As its function was very simple (move and rotate on surfaces of a table), it was decided that its sensing area to be set was fan-shaped only and its recognition events were restricted to object recognition at three directions (right, front and left), which made the setting tasks less bothersome for users. Moreover, one recognition event invokes only one behavior in the current version of RoboTable2. Therefore, the expressive power of RoboTable2 as a programming language was very limited.

If multiple behaviors can be defined for one recognition event and each of them is invoked based on their priorities given by a user, he/she may be able to develop and execute simulation applications with RoboTable2 in more flexible ways. To increase its expressive power, however, it will be needed to implement interaction techniques requesting users with less programming knowledge or experiences to understand more abstract concept (e.g. priorities between behaviors) or conduct bothersome manipulations on RoboTable2. The design of RoboTable2 by considering a trade-off between its expressive power and interaction techniques will be investigated in our future study.

6. CONCLUSIONS

In this paper, we proposed a novel programming environment called RoboTable2 for controlling a tangible robot on a multi-touch tabletop platform, and described programming techniques on the environment. Comparative user studies with a conventional graphical programming environment were conducted to clarify the problem to be solved, issues to be investigated and lessons learned for constructing a design guideline of this programming environment. We also conducted the initial proof-of-concept study and investigated possibilities of RoboTable2 as an application development environment by users themselves. In our future work, we will improve the performance of RoboTable2 through more intensive user studies. Another future work will be to evaluate RoboTable2 in school education with children.

In this paper, all the evaluations were conducted in a single user setting, which needed less complex analyses as compared with a multi user setting, in order to clarify effects of the proposed environment in a simpler situation. It is said that tangible programming can enhance interactions between users and support their collaboration [30]. Therefore, to design and evaluate RoboTable2 so that it can support collaboration between users is also one of our important future works. In the current implementation of RoboTable2, a robot with simple functions is used. Thus, we will investigate if the programming technique in RoboTable2 is applicable to a robot with more complicated functions.

7. ACKNOWLEDGMENTS

This research project received support from Grand-in-Aid for Scientific Research funded by Japanese Society for the Promotion of Science. The authors thank Weiqin, Chen, Shogo Onojima, and Akinobu Niijima for their collaboration and valuable feedback to this project.

8. REFERENCES

[1] Yamaguchi, E., Inagaki, S., Sugimoto, M., Kusunoki, F., Deguchi, A., Takeuchi, Y. Seki, T., Tachibana, S. and Yamamoto, Y.: Fostering Students' Participation in Face-to-Face Interactions and Deepening Their Understanding by Integrating Personal and Shared Spaces, *Transactions on Edutainment II*, pp. 228-245 (2009).

[2] Kanda, T., Hirano, T., Eaton, D. and Ishiguro, H.: Interactive Robots as Social Partners and Peer Tutors for Children: A Field Trial, *Human Computer Interaction, 19*(1-2), pp. 61-84, (2004).

[3] Fischer, G. and Sugimoto, M.: Supporting Self-Directed Learners and Learning Communities with Sociotechnical Environments, *Journal on Research and Practice in Technology Enhanced Learning, 1*(1), pp. 31-64 (2006).

[4] Krzywinski, A., Mi, H., Chen, W. and Sugimoto, M.: RoboTable: A Tabletop Framework for Tangible Interaction with Robots in a Mixed Reality, *Proc. of ACM ACE 2009*, Athens Greece, pp.107-114 (2009).

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

[6] Suzuki, H. and Kato, H.: Interaction-Level Support for Collaborative Learning: Algoblock - An Open Programming Language, *Proc. of CSCL '95*, Bloomington, IN, pp.349-355 (1995).

[7] Horn, M. and Jacob, R.: Tangible Programming in the Classroom: A Practical Approach, *Proc. of ACM CHI 2006*, Montreal, Canada, pp. 869-874 (2006).

[8] Horn, M. and Jacob, R.: Designing Tangible Programming Languages for Classroom Use, *Proc. of ACM TEI 2007*, Baton Rouge, LA, pp.159-162 (2007).

[9] Scharf, F., Winkler, T. and Herczeg, M.: Tangicons: Algorithmic Reasoning in a Collaborative Game for Children in Kindergarten and First Class, *Proc. of ACM IDC 2008*, Como, Italy, pp.242-249 (2008).

[10] McNerney, T.: From Turtles to Tangible Programming Bricks: Explorations in Physical Language Design, *Personal Ubiquitous Computing, 8*(5), pp.326-337 (2004).

[11] Teh, J., Kato, D., Kunieda, K. and Yamada, K.: The Programming of Robots by Haptic Means, *Proc of ACM SIGGRAPH 2008 new tech demos* (2008).

[12] Wyeth, P.: *How Young Children Learn to Program With Sensor, Action, and Logic Blocks, Journal of the Learning Sciences, 17*(4), pp.517-550 (2008).

[13] Schweikardt, E. and Gross, M.: The Robot is the Program: Interacting with roBlocks, *Proc. of ACM TEI 2008*, Bonn, Germany, pp.167-168 (2008).

[14] Smith, A.: Using Magnets in Physical Blocks That Behave As Programming Objects, *Proc. of ACM TEI 2007*, Baton Rouge, LA, pp.147-150 (2007).

[15] Frei, P., Su, V., Mikhak, B. and Ishii, H.: curlybot: Designing a New Class of Computational Toys, *Proc. of ACM CHI 2000*, Amsterdam, The Netherlands, pp.129-136 (2000).

[16] Raffle, H., Parkes, A. and Ishii, H.: Topobo: A Constructive Assembly System with Kinetic Memory, *Proc. of ACM CHI 2004*, Vienna, Austria, pp. 647-654 (2004).

[17] Montemayor, J., Druin, A., Farber, A., Simms, S., Churaman, W. and D'Amour, A.: Physical Programming: Designing Tools for Children to Create Physical Interactive Environments, *Proc. of ACM CHI2002*, Minneapolis, MN, pp. 299-306 (2002).

[18] Ryokai, K., Lee, M. and Breitbart, J: Multimodal Programming Environment for Kids: A "Thought Bubble" Interface for the Pleo Robotic Character, *Proc. of ACM CHI 2009*, Boston, MA, pp.4483-4488 (2009).

[19] Gallardo, D., Julia, C. and Jorda, S.: TurTan: a Tangible Programming Language for Creative Exploration, *Proc. of IEEE TABLETOP 2008*, Amsterdam, The Netherlands, pp. 89-92 (2008).

[20] Kato, J., Sakamoto, D., Inami, M., and Igarashi, T.: Multi-touch Interface for Controlling Multiple Mobile Robots, *Proc. of CHI 2009*, Boston, MA, pp. 3443-3448 (2009).

[21] Seifried, T., Haller, M., Scott, S., Perteneder, F., Rendl, C., Sakamoto, D, Inami, M.: CRISTAL: A Collaborative Home Media and Device Controller Based on a Multi-Touch Display, *Proc. of ITS 2009*, Banff, Canada, pp. 33-40 (2009).

[22] Micire, M., Desai, M., Courtemanche, A., Tsui, K., Yanco, H.: Analysis of Natural Gestures for Controlling Robot Teams on Multi-touch Tabletop Surfaces, *Proc. of ITS 2009*, Banff, Canada, pp.41-48 (2009).

[23] Young, J., Igarashi, T., Sharlin, E.: Puppet Master: Designing Reactive Character Behavior by Demonstration, *Proc. of SCA 2008,* Dublin, Ireland, pp. 183-191 (2008).

[24] Guo, C., Young, J., Sharlin, E.: Touch and Toys, *Proc. of ACM CHI2009*, Boston, MA, pp. 491-500 (2009).

[25] Tse, J., Chan, S., Ngai, G.: An Introduction to the Multi-Modal Multi-Robot (MuMoMuRo) Control System, *Proc. of IEEE SMC 2010*, Istanbul, Turkey, pp.2941-2947 (2010).

[26] Pedersen, E., Hornbæk, K.: Tangible Bots: Interaction with Active Tangibles in Tabletop Interfaces, *Proc. of ACM CHI 2011*, Vancouver, Canada, pp.2975-2984 (2011).

[27] FTIR vs DI (http://iad.projects.zhdk.ch/ multitouch/?p=47/) (retrieved 2010.11.07).

[28] Han, J.: Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection, *Proc. of UIST 2005*, Seattle, WA, pp. 115-118 (2005).

[29] Kaltenbrunner, M. and Bencina, R.: reacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction, *Proc. of ACM TEI 2007*, Baton Rouge, LA, pp. 69-74 (2007).

[30] Horn, M., Solovey, E.T., Crouser, J. and Jacob, R.: Comparing the Use of Tangible and Graphical Programming Languages for Informal Science Education, *Proc. of ACM CHI 2009*, Boston, MA, pp. 975-984 (2009).