Creating a Tabletop Learning Environment Using Physical Robots

Haipeng MI[a*] & Masanori SUGIMOTO[b]
[a]*Interfaculty Initiative in Information Studies, University of Tokyo, Japan*
[b]*Graduate School of Engineering, University of Tokyo, Japan*
*mi@iii.u-tokyo.ac.jp

Abstract: In this paper, we present a novel programming learning environment for kids using physical robots on a tabletop platform. The proposed system supports intuitive multi-touch input on the surface and direct manipulations on physical items, enabling users to learn programming in an intuitive manner. A user study is conducted to verify the usability for applications with a learning purpose. The lessons learned with respect to design guidelines for the proposed learning environment and issues for investigation are discussed.

Keywords: Tabletop, tangible user interface, learning, robot

Introduction

With the rapid development of information technology, many kinds of digital learning materials have been introduced to students. A good example is programming learning activity. Because of its intuitiveness, tangible programming environment has been developed in recent years. An early approach is curlybot developed by Frei et al. [1]. curlybot allows a user to directly drag the robot and repeat the movement the user performed. This simple function enabled an intuitive programming for robots.

Later projects attempted to provide more complex functions rather than defining a motion. For instance, Quetzal is a simple programming tool for elementary school students [5]. By attaching markers, moving blocks and building program chains, the children can learn simple programming concepts such as loop and branch. Another TUI (tangible user interface) programming example is TurTan [3]. By moving programming plates on a tabletop, students can easily define the path of a virtual turtle. Horn et al. compared TUI programming and GUI (graphical user interface) programming methods in an experimental museum exhibition [6]. They counted the successful programming number and code length created by visiting students, and concluded that more students preferred to programming in a TUI programming environment.

In this paper, we present our work of creating a tabletop learning environment using physical robot, called RoboTable. A prototype application named ExploreRobot is developed in the RoboTable environment, which allows kids and programming beginners to easily understand programming basics through direct manipulation and intuitive feedback. A user study is conducted and the result revealed that such a learning environment makes learning of robot programming easier in some aspects such as defining a behavior of the robot and inputting parameters. The proposed method and environment is expected to have applications for many different learning activities.

1. RoboTable Environment

RoboTable in a tabletop platform constructed in our lab [8]. RoboTable integrates DI (Diffused Illumination) [2] and FTIR (Frustrated Total Internal Reflection) [4] so that it can simultaneously recognize multi finger touches and conduct object tracking. We employ physical robots for the RoboTable platform in order to create a tangible interface that has intuitive kinetic feedback. For tracking robot an image recognition library called reacTIVision [7] that can identify a fuducial marker attached to the bottom of each robot is used. A robot is controlled by the system via Bluetooth communications. The tracking scheme and system configuration of RoboTable platform is shown in Fig. 1.

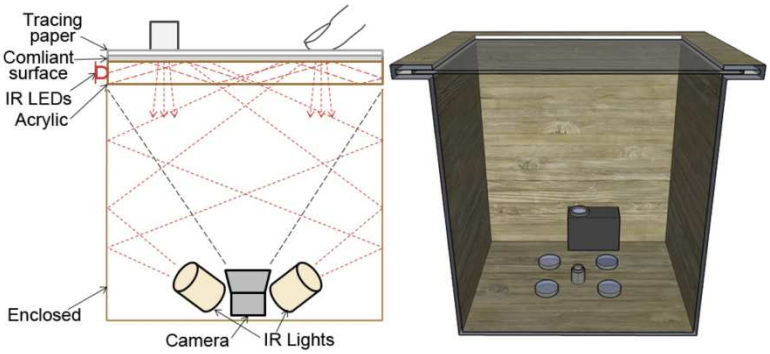


Fig. 1 RoboTable platform: integrated DI and FTIR tracking (left) and the tabletop system configuration (right)

2. Tangible Programming Prototype

2.1 Implementation

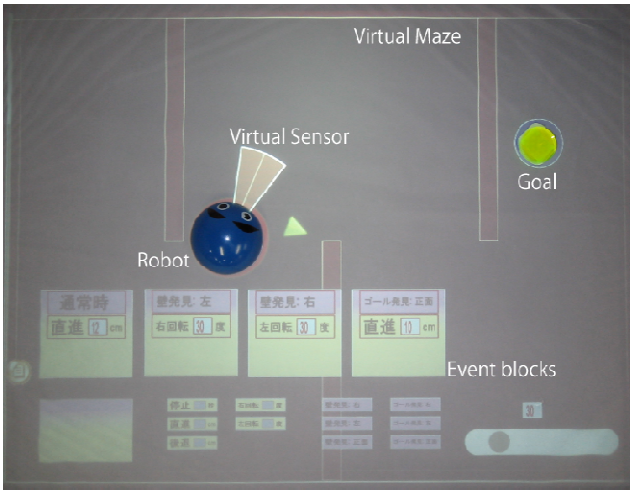


Figure 2. ExploreRobot

Based on the RoboTable platform, we created a prototype application, which is called "ExploreRobot", in order to realize the proposed tangible programming interface. ExploreRobot is a learning assistant application designed for school students and programming beginners. The aim of this application was to help users understand the basic concepts of robot programming. A robot equipped with a virtual sensor is placed in a virtual maze. An acrylic plate is placed somewhere on the table to indicate the goal of the robot explorer. A player has to define actions and set events for the robot in order to make the robot have a complete program to find the path to the goal in a maze.

Fig. 2 illustrates the ExploreRobot prototype, which creates a mixed-reality tangible programming interface that enables basic programming tasks such as finding a specific goal and avoiding obstacles. This programming interface consists of six basic components:

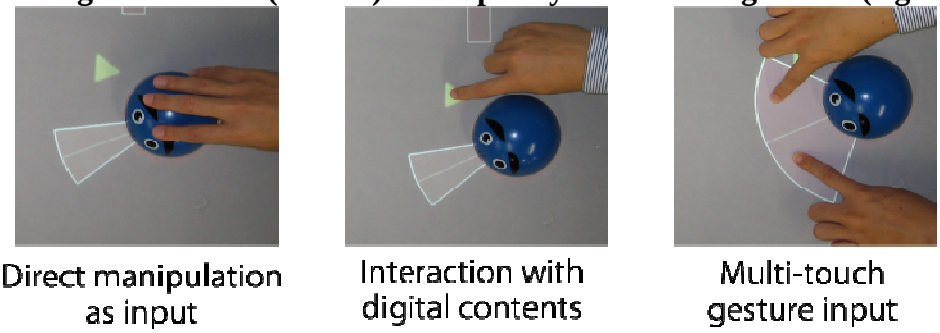
- **Robot**: A real robot is deployed in ExploreRobot. The robot takes two roles in this application. In the programming stage, the robot is used as a tangible indicator, which enables direct manipulation for behavior definition. In the execution stage, the robot is used as a physical programming executor that behaves following defined program.
- **Virtual Sensor**: A virtual sensor is a graphical component attached to the robot and moves with the robot accordingly so that the virtual sensor seems like physically bounded to the robot. The virtual sensor has a fan-shaped detection area with a vertex located at the center of the robot. The fan-shaped area is divided into two parts by the symmetry axis, which coincides with forward direction of the robot. Each part can detect either virtual obstacles or the goal plate and notify the robot of detected object and position (e.g. left or right). Users can adjust the detecting range and width freely.
- **Control Button**: Control buttons are graphical buttons surrounding the robot to provide some specific functions such as record, execute, etc.
- **Event Block**: An event block is a basic programming module, which indicates a series of actions regarding a specific event. All the event blocks compose the whole program.
- **Virtual Maze**: Virtual maze is a set of rectangle obstacles that the robot cannot pass. We created a series of mazes with different difficulty levels so that users can challenge different tasks.
- **Goal**: The goal of the maze is determined by placing a piece of an acrylic circular plate on the table. In each task, one can easily change the position of the goal by direct moving the acrylic plate.

2.2 Programming interface

ExploreRobot supports three input methods for programming the robot.

- **Direct manipulation:** A player can directly manipulate the robot as a tangible token. Direct manipulation is used as a motion input in order to define the robot's behavior (See Fig. 3 left).
- **Interactions with digital contents:** A player can touch buttons displayed surrounding the robot, and drag or move other digital contents such as event blocks. Interactions with digital contents enable players to select functions or build and organize event blocks (See Fig. 3 middle).
- **Multi-touch gesture input:** A player can use multi-touch gestures for some digital contents. Multi-touch gesture input is used for adjusting range of the virtual sensor (See Fig. 3 right).

Fig. 3: Input methods for ExploreRobot: input by direct manipulation (left); interact with digital contents (middle) and input by multi-touch gestures (right)



2.3 Event-driven architecture

The ExploreRobot application prototype uses the even-driven programming architecture. A program is composed by a series of events, and particular behaviors regarding each specific event. An event and the corresponding behavior compose an event-behavior pair, which is the basic component of the event-driven programming architecture.

The first version of the ExploreRobot prototype supports a basic detection event: if a virtual obstacle or the goal enters in the range of the virtual sensor, the detection event is triggered. Each detection event is identified by both the detected object (obstacle or goal) and the position (left, right or front). When an event is triggered, the behavior associated to that specific event is executed subsequently. There is also a special event called "always", the behavior which is always executed if there is no any other event triggered.
To define an event-behavior pair in ExploreRobot is quite simple. A player puts the robot on the table, then the virtual sensor and some buttons appeared subsequently. Moving the robot to simulate a specific event (i.e. seeing obstacle at left), the player is able to press a "Record" button in order to define a behavior sequence regarding this specific event, and then the event block for this specific event is automatically generated (See Fig. 4 left). In the case there is no special event is simulated, pressing the "record" button generates the "always" event block. In order to avoid conflict that more than one event occurs, each event is assigned a priority. In the case of more than one event occurs coincidently, the behavior associated to an event with a higher priority is executed at first, then events with lower priorities. All the priority for each event block is hidden; the system handles priority check automatically. Basically, the "always" event has the lowest priority and the event of goal detection has the highest priority.

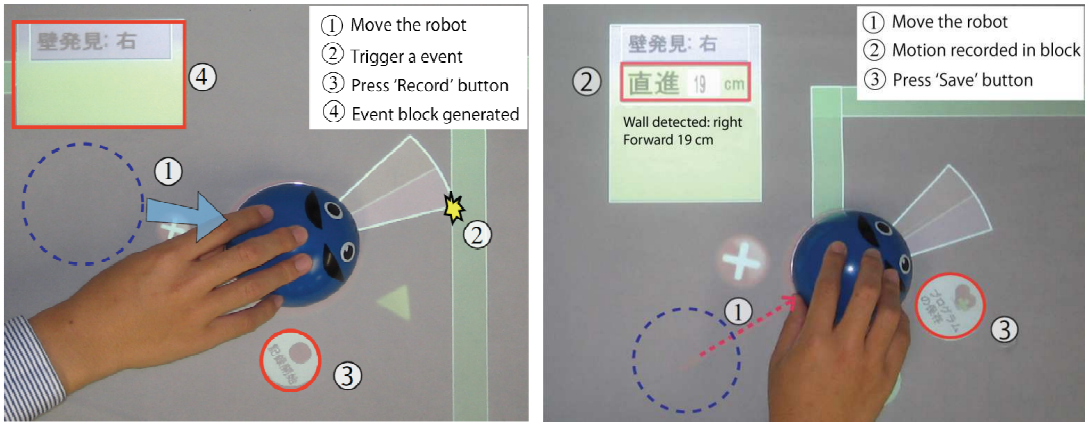


Fig. 4: Generating an event block (left) and defining a motion path for the robot (right)

After an event block generated, the player can directly drag the robot as a motion input to define the motion trace of the robot. A motion trace is composed with a series of basic motions such as moving forward, backward, turning left and right. The motion trace manipulated by the player is automatically decomposed into a motion sequence. Each motion indicator in the sequence indicates the moving direction and specifies the distance (in centimeter) or rotation angle (in degree). Once the player confirmed the motion sequence displayed in the event block, s/he then pushes the "Save" button to finish recording (See Fig. 4 right).

One may ask how if a player wants to modify the motion trace during recording? Actually, the "recording" function does not exactly record everything the player performed. For example, if a player drags the robot 30cm forward, then pushes it 10cm back, the system will not record these two motions respectively. Instead, a motion of

moving 20cm forward will be restored. In other words, the system can automatically determine the motion sequence with minimum length.

When the player finishes all the definition of event blocks, he or she can press the "Play" button to execute the program and see how the robot works. If the robot successfully reaches the goal, the task is finished. Otherwise the player can pause the robot at any time and continue to revise his/her own program.

ExploreRobot programming interface also provides an intuitive management of programmed event blocks. By simply pressing a button, the event blocks switch between shown/hidden states so that the player can easily check the maze map or programming status. If the player wants to delete an existing event block, he or she can just drag that specific event block into a trash box to perform the deletion.

3. Evaluation Experiment

We have conducted a user study to investigate the effect of using the proposed system in programming learning activities. The goal of the experiment is to understand effects on using different programming agent (i.e. physical robot or virtual avatar) and different method (i.e. TUI programming and GUI programming).

3.1 Experiment design

This study used a 2 (agent) × 2 (method) design. For the agent, there were two conditions: physical robot (PR) agent and virtual avatar (VA) agent. The method also had two conditions: graphical programming and tangible programming.

In order to make reasonable comparisons, we also developed a simple graphical programming interface for ExploreRobot. All the components in this graphical programming interface are exactly the same as in the tangible programming interface. However, instead of direct manipulation for parameter input, this graphical programming interface provides several event blocks and action blocks, which can be dragged into event blocks to make a complete event-behavior pair. There is also a slider at the corner of the table that indicates the parameter of the current action block. Sliding the slider can adjust the parameter of the current action block.

3.2 Procedure

A total number of 9 participants (6 male, 9 female, average age: 21.4) were recruited from our university. All participants filled out a questionnaire that surveyed their basic demographics (e.g. gender, age, programming experience). We have confirmed that each participant has no knowledge or very few experience about programming. Participants were given a brief tutorial regarding the basics of the programming environment and manipulations, such as multi-touch, gesture and direct dragging the robot.

Then, participants were given another tutorial regarding programming the robot. The experimenter introduced the event-driven architecture and a sample program, which enables the robot to explore the maze. Once the participant confirmed that he or she could understand the program, the participant carried out the four tasks in four different experimental settings (i.e., one condition was randomly assigned to each task).

After each condition, participants filled out a questionnaire that included self-report items. The order in which participant were assigned to the two agents (ATUI and GUI) was altered. Once the agent was introduced as a real robot or a virtual robot, the order of

method (tangible and graphical) was altered as well. All order assignments were counterbalanced.

3.3 Measure

Evaluation items are scored using self-report questionnaires. The questionnaire includes five evaluation items that measure users' perception regarding their programming process. The five evaluation items are listed below:

- Easy for understanding programming method;
- Easy for setting events;
- Easy for defining behaviors;
- Easy for inputting parameters;
- Easy for imagine the actual movement of the robot.

Each evaluation item uses a 7-point Likert scale, rating from 1 (strongly disagree) to 7 (strongly agree).

4. Result

4.1 Effect on interface

Firstly, a Wilcoxon signed-rank test is conducted in which the interface used in the programming process (i.e. PR or VA) was the independent factor. Table 1 shows the *Z* and *p* values of each item. Fig. 5 compares mean and SD of three items which fulfilled the requirement $p < 0.05$.

Table 1: Effect on agent

Item	*Z*	*p*
Understand programming	-1.890	0.056
Set events	-2.165	0.030*
Define behaviors	-2.111	0.035*
Input parameters	-2.294	0.022*
Imagine robot's movement	-1.769	0.077

From Table 1 and Fig. 5 we can find that the analysis yielded three significant effects of interface and showed that users felt easier when setting events (**PR**: $M = 5.67$, $SD = 0.91$; **VA**: $M = 5.17$, $SD = 1.34$), defining behaviors (**PR**: $M = 5.78$, $SD = 0.94$; **VA**: $M = 5.39$, $SD = 1.20$) and inputting parameters (**PR**: $M = 5.06$, $SD = 1.39$; **VA**: $M = 4.11$, $SD = 1.28$) with a physical robot. This result gave us reliable evidence that users perceived easier in programming processes while using a physical robot for the programming interface.

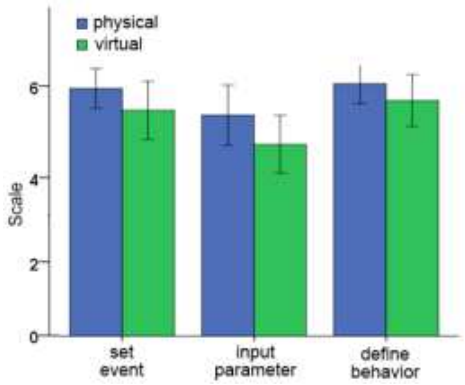


Figure 5

4.2 Effect on method

Next, another Wilcoxon signed-rank test is conducted in which the programming method (i.e. TUI programming or GUI programming) was the independent factor. Table 2 shows the *Z* and *p* values of each item.

From Table 2 we can find that only "easy for imagining robot's movement" meets the requirement of $p < 0.05$. The analysis yielded one significant effect of programming method and showed that users felt easier in imagining robot's movement during programming process (**PR**: $M = 5.72$, $SD = 0.96$; **VA**: $M = 4.56$, $SD = 1.20$)

Table 2: Effect on method

Item	*Z*	*p*
Understand programming	-0.302	0.763
Set events	-0.540	0.589
Define behaviors	-0.722	0.470
Input parameters	-0.894	0.372
Imagine robot's movement	-2.980	0.003*

4. Conclusion

In this paper, we proposed a novel tabletop learning environment using physical robots. A prototype application called ExploreRobot is developed for learning programming basics by defining behaviors for a tangible robot on the tabletop using an intuitive TUI programming method. We have conducted a comparative user studies with a conventional graphical programming environment to clarify the problem to be solved, issues to be investigated and lessons learned for constructing a design guideline of this programming environment. The result revealed that using a physical robot agent and TUI programming method makes programming learning easy. In our future work, we will evaluate the proposed environment 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 learning in a tangible tabletop environment can enhance interactions between users and support their collaboration. Therefore, to design and evaluate the system so that it can support collaboration between users is also one of our important future works. In the current implementation, a robot with simple functions is used. Thus, we will investigate if the programming technique in the proposed system is applicable to a robot with more complicated functions.

Acknowledgements

The authors thank Tomoki Fujita, Weiqin Chen, Shogo Onojima, and Akinobu Niijima for their collaboration and valuable feedback to this project.

References

[1] Frei, P., Su, V., Mikhak, B., & Ishii H. (2000). curlybot: Designing a New Class of Computational Toys. In *Proceedings of CHI '00* (pp. 129–136). The Hague, The Netherlands: ACM Press.

[2] Roth, T. (2007). FTIR vs DI. Retrieved from http://iad.projects.zhdk.ch/multitouch/?p=47/

[3] Gallardo, D., Julia, C. F., & Jorda, S. (2008). TurTan: a Tangible Programming Language for Creative Exploration. In *Proceedings of TABLETOP '08* (pp. 89–92). Amsterdam, the Netherlands: IEEE Press.

[4] Han, J. (2005). Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection. In *Proceedings of UIST '05* (pp. 115-118). Seattle, WA: ACM Press.
[5] Horn, M. S. & Jacob, R. J. K. (2006). Tangible Programming in the Classroom: A Practical Approach. In *Proceedings of CHI '06* (pp. 869–874). Montreal, Quebec, Canada: ACM Press.
[6] Horn, M., Solovey, E. & Crouser, R. (2009). Comparing the Use of Tangible and Graphical Programming languages for Informal Science Education. In *Proceedings of CHI '09* (pp. 975–984), Boston, MA: ACM Press.
[7] Kaltenbrunner, M. & Bencina, R. (2007). reacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction. In *Proceedings of TEI '07* (pp. 69-74), Baton Rouge, LA: ACM Press.
[8] Krzywinski, A., Mi, H., Chen, W., & Sugimoto, M. (2009). RoboTable: A Tabletop Framework for Tangible Interaction with Robots in a Mixed Reality. In *Proceedings of ACE '09* (pp. 107–114), Athens, Greece: ACM Press.