

6th International Conference on Applied Human Factors and Ergonomics (AHFE 2015) and the
Affiliated Conferences, AHFE 2015

Tangible programming environment using paper cards as command objects

Kazuki Tada* and Jiro Tanaka

University of Tsukuba, Tsukuba-shi 305-8577, Japan.

Abstract

This paper introduces a tangible programming environment that uses paper cards as command objects. We have implemented a prototype system called "Sheets". The users can experience a tangible environment without the necessity of specialized devices using Sheets. This system is capable of drawing and moving shapes, and creates loops and branch executions. This system includes paper cards, a webcam, and software. The paper cards are printed descriptions and markers. The markers are captured by the webcam and are recognized by the software. The users can line up the command cards in a specific order to create a program, and the resulting drawing and movements of the graphic are then displayed on the screen. In addition, it is also capable of sensing real-world events such as touch inputs on these command cards, and it is possible to edit the program by writing on the paper cards. For example, the number of loop executions and the values of any variables can be altered. The program can also implement draw commands for custom sketches designed by the users. Sheets can assist in programming using functionalities like conversion to source codes and highlighting. We have performed simple evaluation experiments using this system and collected the opinions of the users.

© 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

Peer-review under responsibility of AHFE Conference

Keywords: Programming Environment; Tangible User Interface; Education.

*Corresponding author. Tel.: +81- 090-7085-9825;
E-mail address: tada@iplab.cs.tsukuba.ac.jp;

1. Introduction

Recently, programming is not just for programmers but has also become an essential part of high school and middle school education. For this reason, there is the necessity for a programming environment targeted towards amateurs with little technical knowledge. Scratch [1] and VISCUIT [2] are examples of such environments, allowing the creation of programs with the help of easy-to-use GUIs. There have recently been a large number of proposals for using a tangible user interface (TUI), such as by Tern [3] and Robert [4], which aim to increase the operability and learning efficiency. There have also been reports on the easier understanding of programming using the TUI instead of a GUI in programming environments [5]. However, the prepare cost of using the TUI for programming environments is high due to the necessity of specialized devices and tools.

We propose a tangible programming environment for learning that uses paper cards as objects (Figures 1a, b, & c). As long as there is access to a printer and a webcam, anyone can start learning programming using this environment. It includes the abilities to convert tangible programs into source codes (Figure 1d) and to highlight the order of execution of the programs (Figure 1e) for easier understanding. Furthermore, a program can be edited by writing on the paper included in the system (Figure 1f).

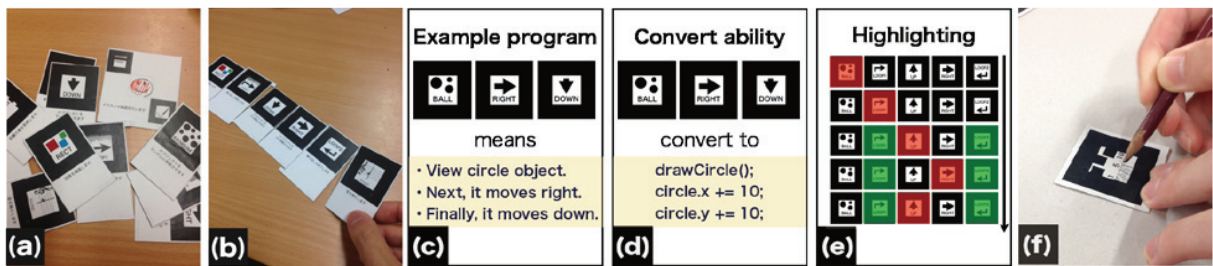


Figure 1. System overview and features.

2. Sheets: Tangible programming environment using paper cards as command objects

"Sheets" is an environment that uses paper cards lined up in order as command objects to learn programming (Figures 2a&2b). By using a paper medium as a command object, it is possible to incorporate handwriting and sketches onto the paper in the process of learning programming. Sheets is capable of drawing and moving shapes, and creating loops and branch executions. Users can line up command cards in a specific order to create a unique program, and the resulting drawing and movement of the graphic is then displayed on the screen. In addition, it is also capable of sensing real-world events such as touch inputs on these command cards. Moreover, Sheets has a function that assists in programming education using different functionalities like conversion into source codes and highlighting.

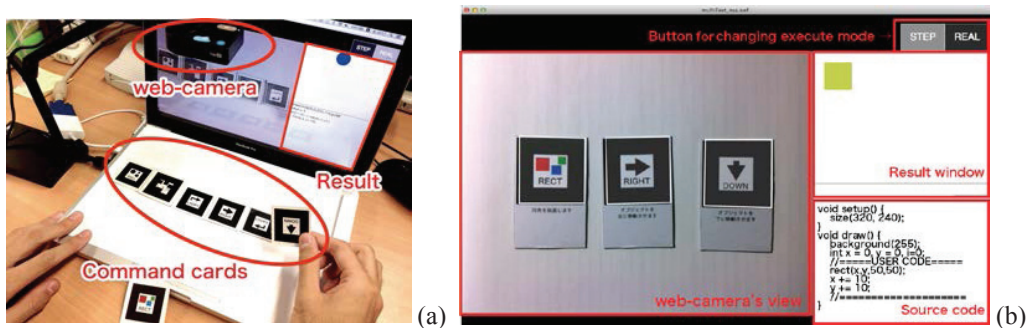


Fig. 2.(a) System structure; (b) Software overview

2.1. Language design

Image drawing or movement is the main objectives of the programs created using Sheets. The users can use commands such as

- Drawing and Moving commands
- Sound commands
- Real-world events commands.

Furthermore, "variable", "loop", and "branch" syntax are prepared. We will describe the content and use of each instruction in the following section.

2.1.1. Drawing and Moving commands

In Sheets, in addition to drawing simple shapes like circles and squares, it is possible to draw an illustration the user painted. Moreover, moving commands are provided to move it vertically and horizontally. The command objects that execute these instructions are shown in Figures 4a and b. The user can arbitrarily move any shapes using a combination of these commands.

2.1.2. Sound commands

The sound command can be used to play a single note of one octave. This command object is in the form of a staff notation. The user can play any sound by drawing the note here. Figure 3c shows how to change the single note for playing by writing on the command object.

2.1.3. Real-world events commands

Sheets can detect real-world events on the command cards and implement them into the program. It is able to detect any touch interaction on the command cards and also the distance between the cards. Thus, the users are able to create a program such as that shown below using this feature.

- Changing shapes by touching command cards (Figure 3d).
- Moving a graphic by moving command cards.
- Selecting a simple musical instrument that responds to touch.

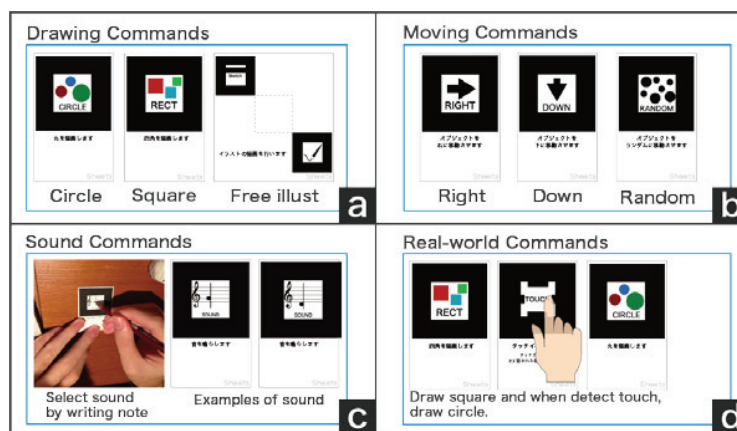


Fig.3. Examples of command objects and descriptions

2.1.4. Syntax commands

In addition to performing the sequential execution as the basic syntax of the programming language, Sheets can be used for the variable syntax, loop syntax, and branch syntax. The command objects to describe each syntax are shown in Figure 4.

Sheets contains a "variable object" used as a value and "variable increase or decrease commands" to increase or decrease the value. "Variable objects" are used in conjunction with such branch commands. The "Variable increase or decrease commands" can be incremented or decremented values of the variable. Switching for the increase or decrease is conducted by filling in a predetermined portion of the command object (Figure 4a).

The loop syntax describes using a "loop start command" and a "loop end command". By sandwiching the commands to be repeated between these two commands, the sandwiched ones are repeated. The number of repetitions is determined by the top of the fill condition of the "loop start command" (Figure 4b).

The branch command can be compared with the values in combination with the variable. The comparison operators and values are determined by writing into the branch command. Figure 4c is an example to compare whether the variable is bigger than 5.

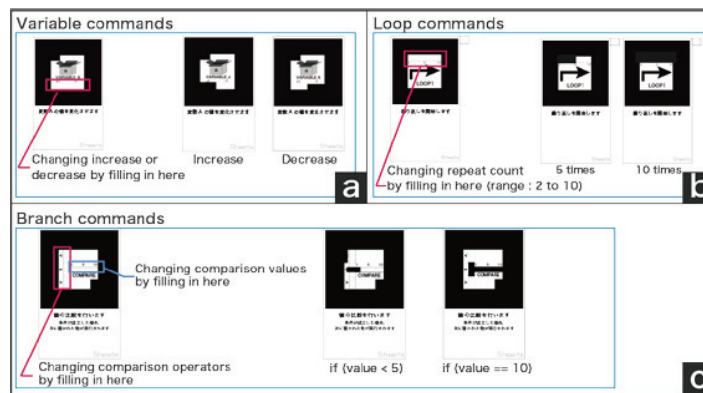


Fig.4. Examples of syntax commands and descriptions

2.2. Learning examples

It is possible to learn the basic concepts necessary for programming such as the sequential execution, looping, branching, and variables, by creating a program that combines the instructions mentioned in this chapter. For example, it is possible to create a "gradually moving animation circle" by combining drawing, moving, and looping (Figure 6a). Furthermore, it is possible to create a program such as "draw a circle if the value of the variable is bigger than 5" by adding the necessary variables and branch commands (Figure 5b).



Fig.5. Sample program

2.3. Implementation

We have already implemented the proposed system. Sheets is made up of paper cards, a webcam, and software. The paper cards contain printed descriptions and markers. These markers are captured by the webcam, and they are then recognized by the software. We used ARToolkit[†] for the marker recognition.

The software developed using Flash (ActionScript3.0) executes the commands and outputs the results. There is no need for any special installation or downloading for the software to run on the Web. The data for the paper cards is also available on the Web.

2.4. Paper cards as command objects

Sheets uses command objects made from paper cards to build a program. The command objects are 64 mm * 40 mm, and an AR marker drawn on one side of them is 38 mm². A description of the instructions for each one is described at the bottom of the markers. The preparation cost of the programming will be reduced by using paper. Moreover, using paper is a cost cutting solution for economically learning programming.

In Sheets, the number of loop executions and the values of the variables can be altered by writing on the command cards (Figure 6a). In addition, the program can implement drawing commands for custom sketches designed by the users (Figure 6b). Furthermore, archives with annotations and comments written right beside each program can be created in books and notebooks.

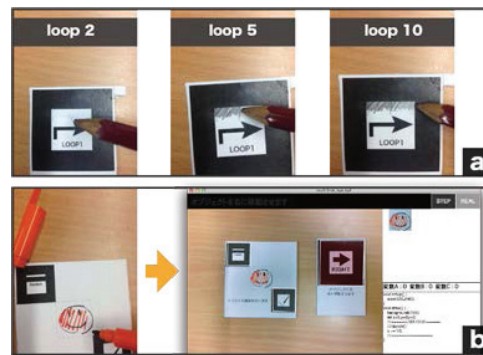


Figure6. Drawing interaction

2.5. Software

This software environment was developed using ActionScript3.0. It is constructed by interpreting the command objects, drawing of the execution results, and using the conversion process of the source code. The execution screen of the software is shown in Figure 3. The software displays the description of the program, the captured images from the webcam, the execution result, and the source codes. In addition, a switching button for the program execution mode is also displayed, and the user presses this button to select whether the Real-time or Step execution mode is to be used. In Real-time execution mode, the execution result is drawn immediately after the user places the command objects. In the Step execution mode, the user can execute the program at any time. In this case, the placed command objects are executed after about one second in order from the beginning. Furthermore, it is possible to visually check the execution order of the program because it is highlighted at the when the command objects are executed.

[†]ARToolkit <http://www.hitl.washington.edu/artoolkit/>

3. Study support features

In this chapter, we describe the learning support functions that are implemented in Sheets. Since Sheets is a programming learning environment for beginners, some mechanisms are provided to aid the new learners in better understanding the programming. We describe the details of each mechanism in the following sections.

3.1. Highlighting order of execution

Sheets is capable of highlighting the command card that is currently being executed, and the current step when executing nested loops and complex conditional branch commands can easily be checked using it. The current step is highlighted in red, and the loop commands are highlighted in green. Each command card in the step execute mode is executed with a one second time delay. In general programming, a change in the execution order with a branch or loop statement is complex for beginners. In this case, the user creates a flow chart, and then, tries to understand it by following it step by step to determine how it would be run. We used the technique to highlight the command objects in Sheets in order to eliminate such a complexity. An example of the visualization is shown in Figure 7.

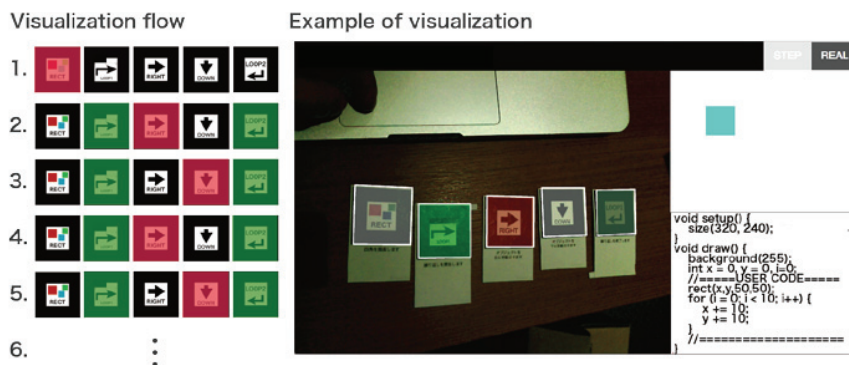


Fig.7.Example of visualization

3.2. Conversion to source code

A program created from lining up the command cards is converted into a source code and then the content is displayed on screen. The output source code is shown as the Processing[‡] language using the original framework. This feature is an attempt that might be able to facilitate better understanding of the description method in programming languages. The source code can be executed by directly copying and pasting the Processing environment. Therefore, it is possible to touch the actual programming language to change simple parts such as the repetition times or circle radius.

3.3. Interaction with paper

We use paper as the command objects in Sheets to improve the ease of learning environment. However, this use is not the paper is not the only medium for the ease of preparation. Sheets incorporates the paper unique interaction into the programming process. Writing onto the command objects and moving the drawn picture is an example. Furthermore, it is possible to capture real-world inputs without needing any special sensors or the like. We tried to make sure the user feels familiar with the programming by including friendly operations such as being able to write on the paper used in Sheets.

[‡]Processing <https://processing.org/>

4. Related Work

4.1. Programming learning support

A variety of techniques have previously been proposed for supporting learning programming. Nigari [6] proposed automatically visualizing the program in order to achieve an intuitive understanding of the program flow. Sato et al. [7] incorporated a visualization system program execution part into the GUI program operation. Scratch [1] and Squeak [8] are environments that can create simple programming using the drag-and-drop of instruction blocks that are displayed on the screen. VISCUIT [2] creates command objects by writing a number or drawing an illustration, and then, creating a program by combining them.

These studies have also attempted programming learning supported by GUI operations such as typing or using the mouse. In our system, we created a programming environment using command objects in the real world, and it is possible to use drawings and sound and other variables as small commands. Moreover, it assists in programming education with functionalities like conversion into source codes and highlighting.

4.2. Programming environment using TUI

Several studies have been proposed to facilitate learning by using the TUI for the programming environment. E-Block [9] and T-Maze [10] use line up blocks containing LEDs or sensors to create an environment in which a program can be created. These environments can be used to program the LED lighting and motor control. tactusLogic [11] and Tern [3] have a programming environment that combines commands and blocks of wood. Turtan [12] can create various visualizations by placing plastic blocks on a table-top interface.

These environments use special sensors or markers for the command object recognition used in the programming. Therefore, in many cases the command objects themselves are expensive and complex. In contrast, our environment uses low-cost and easy-to-use learning tools such as paper as the command object.

4.3. Creative tools using paper

Several previous studies have tried to lower costs by using paper for various types of development and learning. The method in Sketching in Circuits [13] uses conductive tape to create an electronic circuit on paper. Robert [3] made it possible to learn programming by pasting seal-like command objects onto books.

These studies have shown that using paper in a creative environment contributes to the ease of environmental improvement.

5. Preliminary Evaluation

We conducted user studies to validate our approach. We recruited 8 participants (2 female) that were between 18 and 22 years old. Four of the participants did programming, three had little experience, and one had completely no experience. The purpose of the user study was to validate the learning effect and usability of paper cards. We asked the participants to perform programming tasks. They created a program to solve a given task. The following is an example of a task that was given to the students.

- Draw an illustration and move it to the right.
- Change the variables to the indicated values.

The results of the user study showed that the participants who had no experience with programming were able to understand our tangible programming's syntax and created a variety of programs in only a short amount of time. Moreover, using paper cards for the programming improved the learning motivation. In addition, after only our first instructions, the participants were able to build programs without any more help. The participants immediately understood how to program when using Sheets.

6. Future work

We would like to expand the capabilities of the environment. For example, incorporate more features of the paper. In addition, we want to improve the scale of the program that can be created in this environment. We also believe that it is important to conduct more evaluations on the students with no programming experience, such as conducting workshops at elementary schools. We also want to create a new mechanism, such as one that enables multiple people to program together.

7. Conclusion

We presented a tangible programming environment in this paper that uses paper cards as the command objects, and we call this environment "Sheets". Users can use this environment for learning programming without needing any specialized devices. This system consists of paper cards, a webcam, and software.

The users can line up command cards in a specific order to create a program, and the resulting drawing and movement of the graphic is then displayed on screen.

Sheets is capable of drawing and moving shapes, real-world events, and conducting loops and branch executions. Moreover, it is also possible to edit the program by writing on the paper cards. For example, the number of loop executions and the values of variables can be altered. The program can also implement draw commands for custom sketches designed by the users. Furthermore, Sheets can assist in programming using functionalities like conversion into source codes and highlighting.

In addition, we performed a simple user study that showed that Sheets is an easy-to-use environment for learning.

References

- [1] Mitchel Resnick, John Maloney, et.al. Scratch programming for all, *Communications of the ACM*, Vol.52, No.11, pp.60-67, 2009.
- [2] Yasunori Harada and Richard Potter. Fuzzy rewriting: soft program semantics for children, In *IEEE Symposium on Human Centric Computing Languages and Environments*, Vol.1, No.1, pp.39-46, 2003.
- [3] Michael S. Horn and Robert J. K. Jacob. Designing tangible programming languages for classroom use, *TEI'07 Proceedings of the 1st international conference on Tangible and embedded interaction*, pp.159-162, 2007.
- [4] Michael S. Horn, Sarah AlSulaiman and Jaime Koh. Translating Roberto to Omar: Computational Literacy, Stickerbooks, and Cultural Forms, In *Proceedings of the 12th International Conference on Interaction Design and Children (IDC'13)*, pp.120-127, 2013.
- [5] Michael S. Horn, Erin Treacy Solovey, R. Jordan Crouser and Robert J. K. Jacob. Comparing the use of tangible and graphical programming languages for informal science education, In *Proceedings of the 27th international conference on Human factors in computing system (CHI'09)*, pp.975-984, 2009.
- [6] Shinya Cho, Katsuhiko Kakehi, Akira Kawai, et.al. NIGARI system Stairway to Java, In *Proceedings of the IADIS e-Society 2004*, Vol.2, pp.960-965, 2004.
- [7] Tatsuya Sato, Buntarou Shizuki and Jiro Tanaka. Support for Understanding GUI Programs by Visualizing Execution Traces Synchronized with Screen Transitions, *Proceedings of the 16th IEEE International Conference on Program Comprehension (ICPC2008)*, pp.270-273, 2008.
- [8] Luca Cardelli and Rob Pike. Squeak: a language for communicating with mice, In *Proceedings of the 12th Annual Conference on Computer Graphics and interactive Techniques (SIGGRAPH'85)*, pp.199-204, 1985.
- [9] Danli Wang, Yang Zhang, Tianyuan Gu, Liang He and Hongan Wang. E-Block: a tangible programming tool for children, *Proceedings of the 25th annual ACM symposium on User interface software and technology (UIST2012)*, pp.71-72, 2012.
- [10] Danli Wang, Cheng Zhang and Hongan Wang. T-Maze: a tangible programming tool for children, In *Proceedings of the 10th International Conference on Interaction Design and Children (IDC'11)*, pp.127-135, 2011.
- [11] Andrew Cyrus Smith, Heinrich Springhorn, Steven Bruce Mulligan, Ireya Weber and Jackie Norris. tactusLogic: Programming using physical objects, *IST-Africa Conference Proceedings*, pp.1-9, 2011.
- [12] Daniel Gallardo, Carles F. Julia and Sergi Jord. TurTan: a Tangible programming language for creative exploration, In *Proceedings of the 3rd annual IEEE international workshop on horizontal human computer systems (TABLETOP'08)*, pp.412-420, 2008.
- [13] Jie Qi and Leah Buechley. Sketching in circuits: designing and building electronics on paper, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'14)*, pp.1713-1722, 2014.