Tangible Syntaxes: Exploring embodied and tangible I/O programming systems for children

Juan Gabriel Tirado
Universitat Pompeu Fabra
Roc Boronat, 138
08018 Barcelona,Spain
juangtirado@gmail.com

Mónica Rikić
info@monicarikic.com

Giovanni Maria Troiano
Universitat Pompeu Fabra
Roc Boronat, 138
08018 Barcelona,Spain
boboscka@gmail.com

Abstract

In this paper we introduce a theoretical prototype of Tangible Syntaxes, a tangible user interface (TUI) aimed at teaching children about programming and it's basic rules and concepts. Our work sets upon standard approaches in learning programming through tangible and embodied interaction, trying to push forward some of its specific aspects. Our aim is to improve and facilitate the learning process of those aspects of programming that the child might mostly find cumbersome, e.g. syntactic constraints and differences among programming languages and paradigms (e.g. Imperative vs Declarative, Prolog vs Java, Python vs C++, etc…). We propose to solve these issues by developing an intuitive and ready-at-hand interface, which implements the general principles used in the logic of programming, as well as being open to different types of compiling and programming structures.

Keywords

TUI, Tangible Interaction, Education, ICT, TEL, Digital Literacy

ACM Classification Keywords

H.5.2 - User Interfaces, L.1.2 - Learning Objects

Introduction

Digital technologies permeate nowadays nearly every area of our lives through an increasingly diverse set of devices, services and infrastructures. For this reason, few would nowadays dare deny that information and communication technology (ICT) is important for learning and teaching. Either for simple vocational and pragmatic reasons, it is important that young people become skilled and "literate" in ICT, so as to prepare them for life beyond school or for deeper cognitive convictions. "Active Learning" is the suggested approach to take fully advantage of ITC applied to learning domains, and it's now becoming the favorite tendency in technology enhanced learning (TEL) practices (see [6], [7] and [8]).

One of the hot topics in TEL nowadays is how to enhance children's understanding of programming and how to effectively literate them about various programming languages and paradigms (for example, see [18]). As the work of Papert at MIT shows, making programming accessible to children is at the origin of the field of interaction design for children. Starting with Logo [9], many programming tools have been developed with children as the intended users, such as AlgoBlock [17], ToonTalk [5], AgentSheets [13], MicroWorlds [19], and finally the successful Scratch developed at the MIT Media Lab [14].

Tangible Programming for Children

Historically, children have played individually and collaboratively with physical items such as building blocks, shape puzzles and jigsaws, and have been encouraged to play with physical objects to learn a variety of skills. As previously said, it is now commonly agreed that physical action is important in learning, and there is a good deal of research evidence in psychology to support this. Piaget and Bruner showed that children can often solve problems when given concrete materials to work with before they can solve them symbolically ([3], [10]). With careful design of the activities themselves, children can thus solve various problems through manipulation tasks with concrete physical objects. In this sense, tangible interaction and TUIs have proved to be very effective technology to enhance children's learning practices in various domains.

Most of the TUI applications designed for school education would be better classified under Resnick's "digital manipulatives" and "programming construction kits" umbrella; "Kinetic recorders" allow children to teach an electronic toy how to move by repeating a set of guiding motions or gestures, such as Curlybot [4], Topobo [11] or StoryKits [16]. Other devices aim at helping storytelling, by allowing children to record, manipulate and play with audio such as Telltale [1], images and text such as StoryBeads [2], or animated video such as the I/O Brush [12].

Tangible Syntaxes

It is been explained how Object Oriented Programming (OOP) is suitable as a programming paradigm at introductory level for different reasons, i.e. it meets demands for a modern education with powerful concepts (inheritance, polymorphism). Moreover, it seems in line with cognitive processes that are performed in the human brain during perceiving, thinking and problem solving tasks [15]. We believe that, along with the robustness provided by OOP, the creation of a set up in which children can start form physical and symbolic computation, than slowly reaching more abstract reasoning and complexity displayed by the system, this would help them understanding the important principles of programming in an easier and more direct way. This is what we want to implement in Tangible Syntaxes, a tangible interface for learning programming, targeted at children from 7 to 12 years of age. This interface is at the prototyping stage at the moment and it will consist of two different parts:

1. A set of colored tangible bricks and a board, representing the console that the child can use in order to write his/her programs (Figure 1)
2. A physical and/or a digital output, which will perform various types of actions according to the input generated at the programming console.

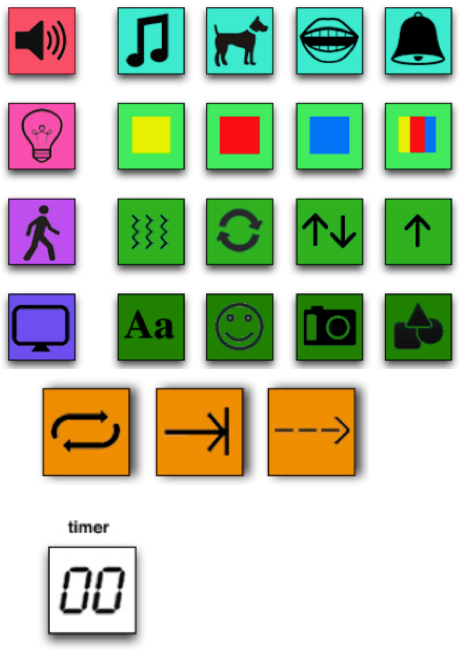

Figure 1: The objects of programming provided by the Tangible Syntaxes interface, i.e. Variables, Definitions, Actions and Arguments

The tangible bricks that the child will use to generate joint series of commands or functions will have in principle the shape of cubes, where each cube has a different image or text printed on top of it and a different color. Each cube belongs to a different type of category, according to the type of programming language that has to be simulated or compiled. For instance, if we were to simulate the type of Processing programming structure, the categories would probably be defined as follows:

- Variables (to be declared at the beginning of the code or inside its segments, i.e. void setup(), void draw(), etc…)
- Objects (for instance myDog = new Dog();)
- Actions (for instance x = x+speed)
- Arguments (i.e. numbers or other kind of events, for instance "10" or "Beep")
- Conditionals (if, while, for, Boolean, etc…)

To compile their codes correctly, the children will have to compose sequences with the aforementioned cubes, respecting the hierarchy established by the syntax and the structure of the program. For instance, putting an argument without defining its relation to a variable or a category will result in no output or displayed error. To make this clear to the child and help them writing their program in the proper way, the bricks will be provided with a small LED light, which will give immediate feedback by flashing GREEN light in case of correct connection or RED light in case of mistaken one (Figure 2).

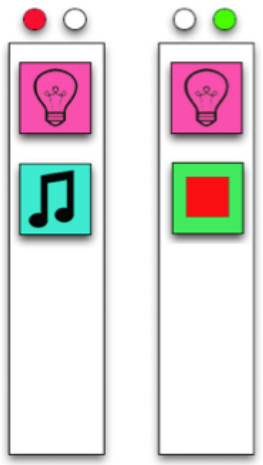
Figure 2: Example of right and wrong connections in Tangible SYntaxes

Once the composition has been completed without "bugs" or mistakes, the child can press the play/compile button, then the output will perform in correspondence to the instructions given at the console. In case a physical output would be used, the objects and the console can be organized so as to simulate and compile accordingly to Arduino's syntax. In this case, the programming objects would remain the same, only the correct sequences and the hierarchy among them may change eventually.

We believe that this system may help children to efficiently and effectively understand the main principles of programming in an active and playful way, as well as implicitly teaching them the bridge that exists between different programming environments and languages.

Discussion and Future Works

As said before, this work is at his prototyping stage and still lacks evaluation and real world implementation, so as to be able to proof its efficiency. However, we believe that the proposed approach could be fruitful and possibly lead to good results in terms of learning impact. Despite the fact that systems as Scratch, Lego Mindstorms[1] or Lego WeDo[2] have proven to be highly effective in teaching children about basic and advanced concept of programming, they have never really shown to be able to create a consistent bridge or metaphor with textual programming environments, which are used in the professional domain. Our aim is to try to fill this gap with a system that can provide this metaphor, as well as investigating possible novel paradigms of embodied and tangible programming. For example, one possible future implementation would be allowing children to program the objects by giving them physical instructions straight on their surface (for example, drawing a circle with the finger on the object would make it rotate in loop, drawing it two times would make it rotate 2 times, and so on). Another possible approach

[1] http://mindstorms.lego.com

[2] http://education.lego.com/en-us/products/

would be the one of using the Tangible Syntaxes system to make children programming each other (for example pretending that a child is the output of the program, and this one will have to follow the instructions received from the console and reproduce them accordingly), so as to understand the programming concepts through real physical actions.

References

[1] Ananny, M. (2002). Supporting children's collaborative authoring: practicing written literacy while composing oral texts. Proceedings of the Conference on Computer Support for Collaborative Learning: Foundations for a CSCL Community (pp. 595–596). International Society of the Learning Sciences.

[2] Barry, B. (2000). Story Beads: a wearable for distributed and mobile storytelling. MIT Masters Thesis.

[3] Bruner, J. (1966). Theory of Instruction. Cambridge, Mass.: Harvard University Press.

[4] Frei, P., & Su, V. (2000). Curlybot: designing a new class of computational toys. ... of the SIGCHI conference on Human ....

[5] Kahn, K. (1996). ToonTalkTM–An Animated Programming Environment for Children (Vol. 7). (Elsevier, Ed.) Journal of Visual Languages & Computing. Leave & Wegner. (1991). Situated learning: Legitimate peripheral participation. Cambridge University Press.

[6] Millner, A. (2008). Supporting Children as they Program to Make Physical and Virtual Objects Interact. Proceedings of the 2008 ACM Interaction Design and Children (IDC) conference.

[7] Millner, A., & Resnick, M. (2005). Hook-ups: How youth learn through creating physical computer interfaces. Media Lab master's thesis. Massachusetts Institute of Technology, Cambridge, MA.

[8] Millner, A., Baafi, E. (2011). Modkit: Blending and Extending Approachable Platforms for Creating Computer Programs and Interactive Objects. Proceedings of the 2011 ACM Interaction Design and Children conference. Ann Arbor, MI

[9] Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas.

[10] Piaget, J. (1973). The child and reality: Problems of genetic psychology.

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

[12] Ryokai, K., Marti, S., & Ishii, H. (2004). I/O brush: drawing with everyday objects as ink. Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 303–310). ACM.

[13] Repenning, A., Ioannidou, A., & Zola, J. (2000). AgentSheets: End-user programmable simulations. Journal of Artificial Societies and Social Simulation, 3(3).

[14] Resnick et al., M. (2009). Scratch: programming for all. Commun. ACM 52.

[15] Schwill, A. (1994). Cognitive aspects of object-oriented programming. IFIP WG 3.1 Working Conference "Integrating Information Technology into Education"

[16] Sherman, L., Druin, A., Montemayor, J., Farber, A., Platner, M., Simms, S., Porteous, J., et al. (2001). StoryKit: tools for children to build room-sized interactive experiences. CHI'01 extended abstracts on Human factors in computing systems (pp. 197–198). ACM.

[17] Suzuki & Kato. (1995). Interaction-level support for collaborative learning: AlgoBlock\—an open programming language. (H. John L. Schnase and Edward L. Cunnius (Eds.). L. Erlbaum Associates Inc.,

Ed.) NJ, USA: In The first international conference on Computer support for collaborative learning (CSCL '95).

[18] Timothy S. McNerney. 2004. From turtles to Tangible Programming Bricks: explorations in physical language design. Personal Ubiquitous Comput. 8, 5 (September 2004), 326-337.

[19] Vincent, J. (2002). MicroWorlds and the integrated brain. Proceedings of the Seventh world conference on computers in education conference on Computers in education: Australian topics-Volume 8 (pp. 131–137). Australian Computer Society, Inc.