Physical Programming: Technology Tools for Kindergarten Children to Program Physical Interactive Environments

Jaime Montemayor
Department of Computer Science, Human-Computer Interaction Lab
monte@cs.umd.edu

Abstract

Children's programming is an important and active research area. But, few are working on ways for young people to author programs that control physical interactive environments. My work addresses this need by developing a physical programming language. This research comes out of our work in developing tools for children to construct room-sized storytelling environments. In this extended abstract, I will describe the motivation for my research, my proposed work and design methods.

1. Introduction

Over the past three years, I have been a part of a research team composed of children and adults. The children are between 7 and 11 years old, and the adults come from disciplines as diverse as engineering, education, computer science, and art [4]. In large part, because of our child design partners, we have come to focus our work on the development of technology that encourages storytelling for elementary school-aged children, and most recently for kindergarteners.

Because storytelling is inherently constructive, the resulting products from our design team have been kits that enable children to create their own stories. Our design goals have evolved from a storytelling robot to a kit that enables children to build physical and interactive story environments [1, 5]. We believe that by giving children the tools to build their own interactive physical environments, they can begin to experience a level of creative autonomy that was previously limited to adults.

There is a wealth of literature on physical interactive environments; active research in visual programming languages for children [e.g., 3, 9]; inquiry into technology for learners; and studies of design methods. But, to my knowledge, there has been little effort that brings together these four areas for the purpose of designing a physical space, in which technology, carefully applied, enhances children's innate learning and exploration styles.

2. Related Work

Making programming easier for the end user has been a focus of at least three categories of researchers. One group attempts to remove the tedium of writing down the exact list of instructions required by computers, from this approach: "watch what I do and repeat this action when it is appropriate." [e.g., 2] This is typically referred to as Programming by Example (PBE) or Programming by Demonstration (PBD). Another group of researchers aim to make programs easier to create and understand by incorporating visual elements into the language (Visual Programming Languages) A third group emphasize the design of visual programming interfaces (Visual Programming Environments) [7]. These systems operate within the confines of the keyboard/mouse and display devices.

An emerging and exciting topic is the discovery and definition of new kinds of human-computer interactions. For example, what would happen if we took away the keyboard and the computer screen? Or, what if the environment surrounding the user IS the input and the output devices? [e.g., 8] Current tangible devices demonstrate the potential usefulness of a more natural and direct manipulation of physical objects, but they do not allow users to construct their own rules for interaction.

Papert's mechanical turtle [11], the Curlybot [6], and Tangible Computation Bricks [10], are three rare examples of programming systems by tangible manipulation of real physical objects.

3. Proposed Work

Based on our formative research on StoryRoom, StoryKit [1], and Programming for Children, I propose to design and develop a proof-of-concept prototype to demonstrate that programming for young children can be a concrete and physical activity. In particular, kindergarten students can use direct physical actions to author, or program, simple interaction rules of storyrooms.

This physical programming language can generate augmented deterministic finite state machines *StoryRoom*

= (I, E, Q, Σ, δ, q_0, F), where I and E are the sets of sensors and effectors in the environment, respectively; Q is a finite set of states, each state q Q contains the instantaneous values of all devices in E; Σ is the alphabet. Each alphabet character is a set of ordered pairs, in which the first element of the pair is a sensor, and the second element a valid sensor value.

The alphabet Σ can be extremely large! But, what if kindergarten children only care about a very narrow set of events? For example, they might only care about the current values from every sensor, instead of the values from some subsets of I. This greatly reduces the effective size of Σ. So, knowledge about relevant physical events is important for a system to handle a physical storytelling environment for children. Unfortunately, unlike traditional keyboard/mouse interactions that are well understood (e.g. *mouse down*, *mouse over*, *key down*), to my knowledge, no one has developed this list of standard physical events for ubiquitous computing environments.

This then, is the first goal of my proposed research: Develop a list of standard physical events for storytelling.

The next step towards the implementation of this *StoryRoom* DFA generator is a method for children to define the transition function. This is the conditional branching feature of a programming language. However, we do not know the natural physical gestures and manipulative movements children would employ to define this *if-then* relationship.

So, this is the second goal of my proposed research: Define the direct physical manipulation gestures (syntax), that are computationally recognizable, for children to specify state transitions.

Finally, how powerful should physical programming be? So far, research with our child designers suggest that this augmented DFA may be sufficiently powerful and useful for kindergarten children. Therefore, for now, I will not investigate whether physical programming can support other elements of programming languages such as loops and variables.

4. Acknowledgments

This work has been funded by the European Union’s i3 Experimental School Environments initiative, DARPA, and the Institute for Advanced Computer Studies. I would also like to thank Allison Druin, Jim Hendler, my colleagues in the Human-Computer Interaction Lab, the PLUS Lab, and the children design partners on our team.

5. References

[1] Alborzi, H., Druin, A., Montemayor, J., Platner, M., Porteous, J. Sherman, L., Boltman, A., Taxén, G., Best, J., Hammer, J., Kruskal, A., Lal, A., Plaisant-Schwenn, T., Sumida, L., Wagner, R., and Hendler, J. (2000) Designing StoryRooms: Interactive Storytelling Spaces for Children. In Proceedings of *Designing Interactive Systems* (DIS-2000), ACM Press, pp 95-104.

[2] Cypher, A., Halbert, D. C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B. A., and Turransky, A. (eds.) *Watch What I Do: Programming by Demonstration*. Cambridge, MA: The MIT Press, 1993.

[3] Cypher, A., and Smith, D. (1995) Kidsim: End-user programming of simulations. In *Human Factors in Computer Systems: CHI 95*. ACM Press, pp 27-34.

[4] Druin, A. (1999) Cooperative inquiry: Developing new technologies for children with children. In *Proceedings of Human Factors in Computing Systems (CHI 99)*, ACM Press.

[5] Druin, A., Montemayor, J., Hendler, McAlister, B., Boltman, A., Fiterman, E., Plaisant, A., Kruskal, A., Olsen, H., Revett, I., Plaisant- Schwenn, T., Sumida, L., and Wagner, R. (1999) Designing PETS: A Personal Electronic Teller of Stories. In *Proceedings of CHI’99*, ACM Press, pp 326-329.

[6] Frei, P., Su, V., Mikhak, B., and Ishii, H. (2000). curlybot: Designing a new class of computational toys. In *Proceedings of Human Factors in Computing Systems*, pages 129–136. ACM Press.

[7] Green, T. R. G. (1995) Noddy's guide to ... visual programming. In *Interfaces, The British Computer Society, Human-Computer Interaction Group*. 1995.

[8] Ishii, H. and Ullmer, B. (1997) Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms, in *Proceedings of CHI* ‘97 (Atlanta GA, March 1997) ACM Press, pp 234-241.

[9] Kahn, K. (2000). Generalizing by removing detail: How any program can be created by working with examples. Available at http://www.animatedprograms. com/PBD/index.html.

[10] McNerney, T. S. (2000). Tangible programming bricks: An approach to making programming accessible to everyone. Master’s thesis, MIT Media Lab.

[11] Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas*. Basic Books, New York.