Handcrafted Physical Syntax Elements for Illetterate Children: Initial Concepts

Andrew Cyrus Smith
African Advanced Institute for Information & Communications Technology
PO Box 395
Pretoria, 0001 South Africa
acsmith@csir.co.za

IDC '08, June 11-13, 2008 - Chicago, IL, USA
ACM 978-1-59593-994-4

Abstract
We present two technology-augmented physical materials that illetterate coders can sculpt for use as physical syntax elements in a tangible early-programming learning environment. Two physical coding sequences are given. We conclude with the listing of further work required.

Keywords
Clay, craft, interface, illetterate, program, syntax, sculpt, stone, tangible.

ACM Classification Keywords
H5.m. Information interfaces and presentation (HCI): Miscellaneous.

Introduction
There have been many attempts to make computer programming accessible to more people [6]. Our research is aimed at introducing illetterate (a narrow definition of non-literacy [8, p11]) children to the topic of programming. Our physical syntax, although simple, has the benefit of eliminating syntax errors such as those encountered in most text-based or icon-based programming environments. Manipulating physical objects to form a sequence of steps does not require the user to operate a computer nor the ability to read. It has been reported that the computer screen limits the human experience [1 p3]. In contrast, using

physical objects broadens the human experience over that offered by the two-dimensional world of computer keyboard, mouse and screen. Although we live in a three-dimensional world, current computer interaction technologies limit us to two-dimensional interaction [1 p4]. We are interested in the user's innate ability as well as already-learned skills to express abstract concepts by manipulating real-world objects.

The contributions of this work are the initial concept designs for end-user-crafted physical programming syntax elements for use by illetterate users.

Advantages

1: One advantage of using physical devices over abstract symbols as program representations is that the physical devices change with usage. Over time and use, physical input devices develop a character of their own. They get worn through continued manipulation and it becomes clear which are the most popular [1 p20-21], potentially also providing a cue to the next user on how to use them. 2: Another advantage is that the device's affordance can be changed and improved by the user. The user can shape the device to represent familiar objects, or new objects that represent abstract ideas. 3: Using physical artifacts as input devices, we can move around and others can participate in the logic construction. The computing device is no longer for the use of one person at a time, but can support co-located team collaboration. 4: The advantage of a child shaping its own interpretation of a programming command is that it represents the child's unique mental abstraction of that command. 5: It also eliminates the stereotyping often found in toys; some toys are aimed at girls and others at boys. Now they can each construct to suit their own preferences in shape and color. 6: For a child to learn best, it should be done with the child's culture in the centre of the learning program [2 p3]. Similarly, we believe a child comprehends new ideas best if they relate to familiar objects. The child can construct representations of familiar objects, reducing the cognitive load of first internalizing the given object or symbol and then translating that to the action it represents. This is one of the leaps required by a text-based programming environment: Letteracy is required from the programmer. The programmer has to be familiar with the symbols used as well as the combination that make up the words used in the syntax; words such as 'void', 'begin', and 'for'. Using physical objects constructed by the user eliminates the need to make this association. It exploits the intrinsic representation which the crafted object holds for that particular child. 7: In addition we learn through our senses [3 p55]. Constructing your own physical representation of the action is a creative experience, using touch senses, and visual perception through the eyes of the creator. Letteracy requires the ability to interpret text symbols and associate meaning with them. But if the child hasn't had any prior exposure to these words then they are meaningless, they are not comprehensible [3 p54]. 8: By constructing the physical syntax objects herself, the objects contain their meaning as meant by the creator. Exploiting the user's innate abilities requires no or very little training. While exploiting already-learned skills requires little re-training for the new application of existing skills [[9] as referenced by [5 p2467]]. For example, to shape clay to represent the response required from an embedded system.

Related Work

Some research into controller objects are aimed at children and use symbols only: Story telling [12] consists of cubes that use pictures on the face of cubes. ActiveCubes [7] does not make use of any symbols or text and is mainly used for assessment and not learning by the user. Electronic Blocks [13] makes use of symbols on the sides of plastic blocks. AlgoBlock [11] utilizes a combination of symbols and text. Tern [4] consists of text on 2-dimensional wooden blocks. The GameBlocks [10] programming environment consists of input devices, interpreting electronic circuitry, and output devices. GameBlocks uses icons (fig 1b,c) on top of the cubes to represent the cube's functionality. A coded sequence is shown in fig 1d. Our research extends GameBlocks to include hand-crafted, physical coding objects.

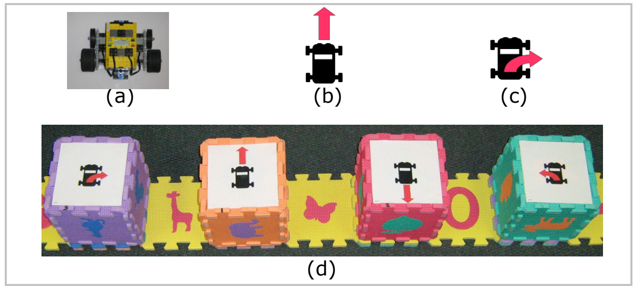

figure 1: The motorized toy car (a) is an output device for the program sequence. Icons representing "*forward*" (b) and "*right turn*" (c) motions. (d) An example of a four-part programming sequence using GameBlocks cubes placed on the programming mat – "*turn right*", "*go forward*", "*go backward*", and "*turn left*". The sequence is interpreted from left to right.

Implementation

The simple magnetic encoding concept used in the GameBlocks cubes can be extended to natural non-ferrous materials. We investigated two materials; soft stone and air-drying sculpting clay. The soft stone was shaped using hand tools (fig 2a,b) and the modeling clay by hand (fig 2c). These serve the same function as the icons-and-cube combinations of fig 1. The sensor slab is crafted from stone and contains four magnetic switches (fig 2d), one in each corner of the recessed square. The arrow crafted from stone contains a single permanent magnet at the front. The clay car also contains a single magnet at the front. Depending on its orientation, placing the input device into the tray closes one of the switches. Trays (fig 3) are sequentially interrogated by an electronic circuit which executes the instructions associated with each orientation.

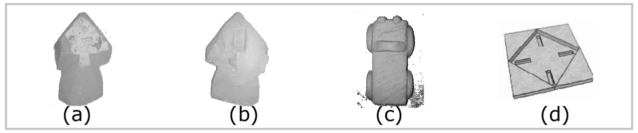

figure 2: (a) Top view of the arrow crafted from stone. (b) Bottom view, with the magnet cavity visible. (c) Sculpted clay car. (d) Sensor slab.

Although the effort in crafting artifacts from the soft stone is significantly more than using the modeler's clay, stone has the advantage of being more durable than the clay. The risk in using a soft stone is much higher than using clay as the stone can break apart along natural weak lines. In contrast the clay is homogenous and can be recycled when the crafter makes a mistake. The recessed area in the sensor

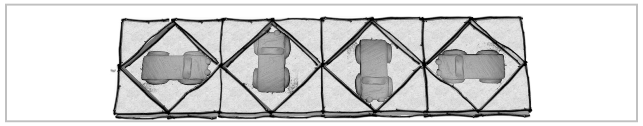

figure 3: The same program as in figure 1d.

slab improves the affordance of the system design as it allows for easier alignment of the input device and the programming tray.

Future work

A fifth magnetic sensor can be added in the centre of the sensor slab. This would allow for either one or two magnets to be embedded in the inserted artifact, doubling the usable function encoding space from four to eight. Alternatively, the increased coding space could be used to assign properties to the coding objects, such as speed in the case of the clay car. We would also like to investigate, with children as design partners, the use of other materials as input artifacts, such as wood and paper.

Conclusion

We have given the potential benefits of using custom handcrafted physical syntax objects to introduce illetterate people to the abstract thinking required for computer programming. Examples of objects given include the use of soft stone and clay.

References

[1] Gershenfeld, N. When Things Start to Think, Henry Holt and Co., Inc., 1999.

[2] Grove, M.C. and Hauptfleisch, H.M.A.M. Stepping stones- a school readiness programme, Part 1 Teacher's manual. 1985. De Jager-HAUM Publishers.

[3] Hannaford, C. Smart moves- why learning is not all in your head. 2005, Great River Books.

[4] Horn, M.S. and Jacob, R.J.K. Tangible programming in the classroom with tern. In CHI '07 extended abstracts on Human factors in computing systems, ACM Press (2007), 1965-1970.

[5] Jacob, R.J.K., Girouard, A., Hirshfield, L.M., Horn, M.S., Shaer, O., Solovey, E.T. and Zigelbaum, J. Reality-based interaction: unifying the new generation of interaction styles, CHI '07 extended abstracts on Human factors in computing systems, ACM Press (2007), 2465-2470.

[6] Kelleher, C. and Pausch, R. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers, ACM Comput. Surv., 2005, vol 37,83-137, no 2.

[7] Kitamura, Y. and Kishino, F. Real-time 3D interaction with ActiveCube, In CHI '01 extended abstracts on Human factors in computing systems, ACM Press (2001), 355-356.

[8] Papert, S. The Children's Machine: Rethinking school in the age of the computer, Basic Books, 1993.

[9] Reed, S.K. Transfer on trial: Intelligence, Cognition and Instruction. In Singley, K. and Anderson, J.R. eds. The Transfer of Cognitive Skill, Harvard University Press, Cambridge, MA, 1989, 39.

[10] Smith, A. C. Using magnets in physical blocks that behave as programming objects, Proc. Tangible and embedded interaction, 147-150, ACM Press (2007).

[11] Suzuki, H. and Kato, H. Interaction-level support for collaborative learning: AlgoBlock-an open programming language, CSCL '95: Computer support for collaborative learning, 1995, 349-355.

[12] Tang, T. S. Storytelling Cube: A tangible interface for playing a story, 2006. Accessed online 10 April 2008 http://code.arc.cmu.edu/lab/upload/new_poster2.0.pdf

[13] Wyeth, P. and Wyeth, G. Electronic Blocks: Tangible Programming Elements for Preschoolers, In Proc INTERACT 2001, 2001.