Rock Garden Programming

Programming in the physical world

Andrew Cyrus Smith
CSIR Meraka Institute and University of South Africa
Pretoria, South Africa
acsmith@csir.co.za

***Abstract*—The Internet of Things (IoT) holds the promise of improved programmatic user control over domestic appliances. The developed world dominates the design of programming environments, assuming letteracy and computer literacy on the part of the programmer. In developing areas, this assumption raises the bar for novice programmers and especially pre-school children from differing socio-economic and ethnic backgrounds. In general, our research aims at developing a programming environment that does not require either computer literacy or letteracy from the programmer, thereby affording the novice user the opportunity to control appliances connected to the IoT. A tangible environment can potentially remove both of these obstacles. Also of special interest to our research is giving the programmer the opportunity to craft her own tangible objects, giving the objects personalised properties. To this end we report on how well pre-school children from differing socio-economic and ethnic backgrounds were able to use a tangible programming environment consisting of direction indicator arrows, crafted from soft stone. In addition we provide examples of further objects that could be used as programming objects. Finally, we show the steps involved in constructing a tangible program with which the user can hypothetically instruct a lawn mower though the IoT.**

Keywords—Internet of Things; tangible program

I. INTRODUCTION

Previous work reported on using materials found around the home to serve as programming objects, such as carved natural rock [10]. This can be done when these everyday objects are modified and used along with electronic circuits. Such a configuration can then be used to control other objects, such as motorised toy cars. In this paper we report on tests conducted with pre-school children to determine the feasibility of using modified everyday objects as programming agents, as well as the possible application of the physical objects in creative gardening when combined with the Internet of Things (IoT).

The majority of the world's population is excluded from the creative pastime of programming. Programming traditionally means having access to a computer and being computer literate. In addition, knowledge of the programming environment is also required. The latter requires from the user the knowledge of opening an editor, entering code by typing, compiling and executing the code. If the compiler reports errors, the user has to find and repair them before attempting the compiling cycle again.

Some programming environments are completely inappropriate for the casual programmer, adding so much code overhead (sometimes also automatically added, in addition to what the user writes) to a simple program that the coder can easily get overwhelmed. Environments such as these are well suited for the experienced programmer, but not the novice. Examples include Visual Studio [1] and Delphi [2].

Other programming environments reduce the code that is visible to the programmer, keeping much of the underlying operation hidden. An example is the Processing [3] programming environment.

Having considered the method of programming, we now discuss aspects of concretising the results of an executing program.

Output of a program is usually confined to either the computer screen or to processes invisible to the user (for example, updates to a database). To the novice programmer this limited output has limited value. Of greater value would be concrete results observable in the physical world. Papert has extensively discussed this in his writings on Logo Turtle. See for example [4].

Programmable devices are becoming readily available. But the average person is severely restricted in what he can do with the intelligent devices already on the market, such as programmable lawnmowers and home vacuum cleaners [5]. In order to make extensive use of these products' abilities, the user has to have access to a computer, be computer literate, and understand the programming environment (IDE). These requirements limit the number of people who will actually attempt programming such devices.

Our aim is to simplify the programming experience for the novice by providing a tangible environment for coding, and observing the results in a concrete manner. This environment also does not require a computer or computer literacy on the part of the coder. Such an environment limits the complexity faced by the user by providing reduced capability. Instead of being a do-all environment, it is an environment with limited ability focused for use in controlling devices in- and around the home.

The system we describe here implements only four motion commands, plus a delay command. The commands do not take any parameters, and no compiling is necessary; the user simply constructs the sequence and initiates its execution by turning on the power.

II. Prior work

Most tangible programming environments rely on the use of text to convey the meaning of the programming elements. Some environments have made attempts at eliminating text to some extent. The challenge is in representing both objects and actions using only static pictures. Not surprisingly, these have been described as nouns and verbs in the context of tangible user interfaces (TUI's) [6].

Fernaeus et al. reports on the potential of using contextual signs in programming. They borrow this notion from comic books [7]. Fig. 1 (left) illustrates the use of arrows, a ghost image, and speed lines to indicate motion. reacTable [8] (Fig.1, middle) uses symbols typically used in engineering and the music industry. Users from these backgrounds intuitively know what the effect will be when using tangibles marked with these symbols. Horn et al. (Fig. 1, right) combine text and shapes to convey the meaning of the tangible [9].

Fig. 1. Indicating motion using image attributes, icon representing a saw wave signal generator, and TUI combining text and shape.

III. System description

Our tangible programming environment consists of three elements, plus the object that will be controlled (Fig. 2). These three elements are a number of similar contextual objects (Fig. 3), a state machine, and sensing tiles. The object being controlled can either be a motorised toy car, a remote controlled lawnmower, a programmable vacuum cleaner, or any one of many programmable consumer items.

Arrows carved from natural stone serve as programming elements (Fig. 3).

To sense the orientation of the input, low cost magnetic switches are embedded into programming squares, called Sensing Tiles. These tiles are connected to a low-cost state machine that interrogates inputs from the magnetic sensors. Each tile has a number of magnetic sensors, called reed switches. These close when in close proximity of a magnet. As an example [10], each tile contains three sensors, allowing for sensing seven different magnet configurations.

Objects placed on the tiles have magnets embedded in them, is a position which is close to the base so as to activate the switches in the tile when the object is placed on top of the tile. When activated, the state machine interrogates the switches to determine in which orientation the object on the tile has been placed.

The tiles are laid in sequence, representing the sequence of program execution. A one-way link sends instructions from the state machine to the device which in turn executes the commands. An application example is the control of a remote controlled lawn mower. The lawn mower follows the path programmed using the rocks. In this way patterns may be mowed in the lawn, similar to those found at sporting grounds.

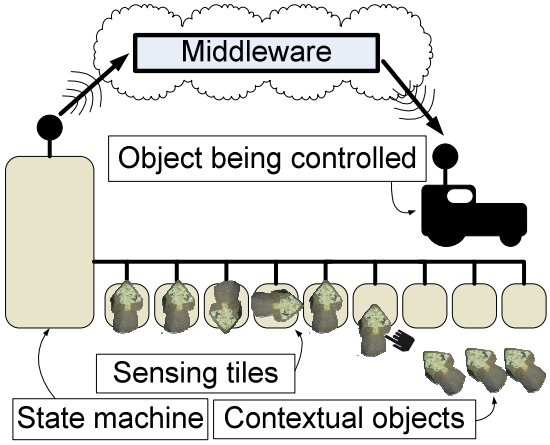

Fig. 2. The diagrammatic representation of the Rock Garden Programming Environment.

Fig. 3. Arrow carved from soft rock with attached magnets (not visible).

IV. Evaluation

Our aim was to determine if the rocks, together with the programming tiles, could serve as a tangible programming environment for illetterate children. This implied that no reading ability could be expected from the target group. The programming objects were designed to be intuitive due to its arrow shape. Because this was a novel system, we had to introduce it to the participants by explaining and demonstrating its operation.

We evaluated the programming environment with the assistance of children from two schools (Fig. 4). The evaluation was conducted at two locations in South Africa, with two separate groups from different socio-economic backgrounds. One group was from an affluent suburban area, and the other from a township. The tests were conducted at two pre-school centers. The children were of mixed gender and

their ages ranged from 5 to 6 years. All the children were affluent in English, the language in which the instructions were given. In total 27 children (13 boys, 14 girls) participated in the evaluation.

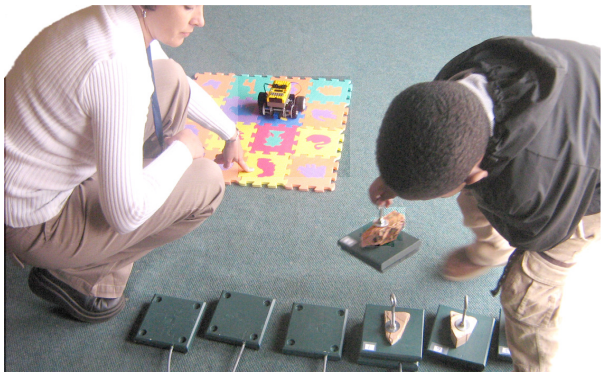

Fig. 4. Evaluating the use of contextualized objects as programming agents. Five programming tiles are visible in the foreground. Two programming rocks are already in place on the tiles, and a third rock is about to be placed. In the background the motorized toy car can be seen on the execution mat. The instructor is indicating to the participant where the car should move to next.

The evaluation was part of a tree-part test aimed at determining the visual perception skills of the children [11]. Programming was the third part of the tests. The other two parts were the matching of two-dimensional to three-dimensional objects, and matching the direction of the rocks to arrows drawn on paper. Here we only report on the results of the programming evaluation.

The task given to the children was to move the motorized toy car to specific positions. This could be achieved by placing the programming rocks in the required orientation onto the tiles.

For these tests the rocks were mounted onto wooden blocks and a metal bolt attached. The bolt served two purposes. First, it served as a means to secure the rock to the wood. Second, it served as a convenient handle in moving and orienting the rocks onto the tiles.

V. Results and Discussion

Initially all the movements (forward, back, left, right) were quickly grasped. However, once the orientation of the toy car had changed after a left or right turn, it was clear that the cognitive load experienced by the participant had increased.

Other workshops we have conducted with adults indicated that adults face the same increase in cognitive load when the car's orientation has changed. However, this did not deter the participants from renewing their efforts to solve the problem of changing orientation. They were determined to understand what was taking place and construct the correct sequence in order to achieve their goal.

Challenges posed at young participants were much simpler to those given to the older children and adults at the science workshops. At the science workshops the children observed how each participant tried to solve the challenge given. This resulted in a steady improvement in finding the solution as a participant took over from the previous. Our sessions with the pre-school children did not allow for this learning; each participant was alone with the researchers when the task was executed. No learning could take place by observing previous attempts made by others. We followed this approach in order to determine the applicability of the programming environment for measuring children's visual perception. In contrast, at the science workshops, the goal was rather to provide an experimental programming environment in a group context.

From our observations of both children and adults participating at science festivals workshops, as well as the directed pilot studies such as those already reported [11], we can conclude that the programming environment in this paper can be viewed as a truly interesting microworld [12] in which people can use mathematics (direction, distance), think about it (change in orientation), and play (tangible input and tangible output).

As mentioned earlier in the paper, the test was part of a larger three-part test by which we aimed to determine the appropriateness of tangible programming objects in measuring a child's visual perception skills. There is a view amongst postmodernists that biological age alone should be considered. Differences in cultures and societies also have an effect on measurements [13]. For this reason we conducted the tests amongst two groups from differing socio-economic backgrounds.

Although there was a significant difference between the two groups when we conducted the matching and orientation tests, we did not find any significant difference in their ability to master the use of the tangible programming environment. Familiarity with the programming objects (having been fashioned from natural stone) has been offered as an explanation for this observation [14].

VI. Prospects

In our research we continue to contemplate alternative tangible programming objects. Here we give two examples of contextual objects that can serve as programming objects, and these being bicycle model and a canoe model (Fig. 5). In the case of the bicycle taxi, the combination of bicycle and cyclist represent the computing device, with the passenger representing the sequence of program instructions because "she gives verbal instructions to the cyclist".

In the case of the canoe, the canoe represents the computing device and the rowers represent the program instructions. This is because "the rowers have the ability to decide where, and how fast, the canoe should move".

We envisage a system where parameters are represented by the properties of the programming objects. For instance, the passenger on the bicycle can represent a drowsy person to indicate a slow pace. A person with hands waving in the air indicates high speed. The number of rowers in the canoe can indicate the speed/distance of travel.

Fig. 5. Items that have been sourced in Uganda could represent motion and direction: Bicycle with two people and a goat; Canoe with three rowers.

VII. Lawn Art

The following is a fictitious example of using hand crafted rocks in the garden as programming objects. In this scenario garden stepping stones (Fig. 8) have magnetic sensors embedded. By placing the programming objects (Fig. 3) in the desired orientation onto the stones, a program is constructed. If these instructions are sent to a remote-controlled lawnmower, the result will be visible as a pattern mowed in the lawn.

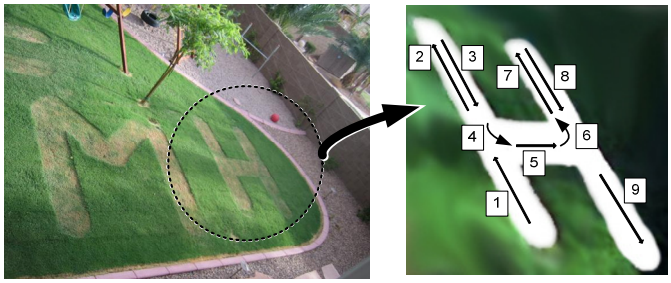

Fig. 6. Example of utilizing the lawn as a canvas. Mowing the lawn along certain paths creates recognizable patterns. An enlarged view of the "H", showing the nine steps in making it.

Fig. 6 shows how a program is constructed in this scenario: A series of simple movements are converted to orientations (Fig. 6, right). The following sequence is determined through inspection: (1) forward, (2) forward, (3) back, (4) right, (5) forward, (6) left, (7) forward, (8) back, (9) back. The programming objects are then placed in sequence (Fig. 7). When the program is executed, the result is the letter "H" which is visible in the lawn.

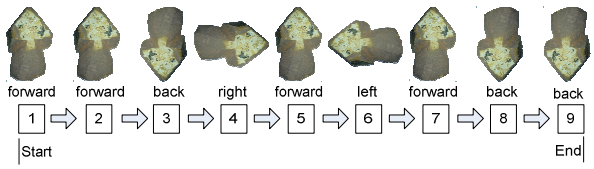

Fig. 7. The constructed program that corresponds to the steps shown in Fig. 6.

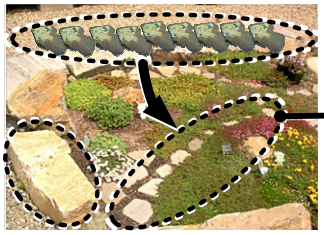

Fig. 8. The rock garden before the program is constructed. In the foreground, a row of stepping stones serve as programming tiles. Each tile has been embedded with a sensing device. The large rock on the left contains the electronics and communication device to the remote controlled lawn mower.

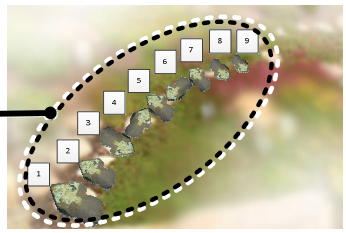

Fig. 9. (right). The constructed program in the rock garden context.

VIII. Zen Gardens and Rock Art

In conclusion, we present two familiar artefacts which we will be investigating in future research. These are the miniature Zen garden and the hand-painted stones (Fig. 10, 11).

Fig. 10. A smaller rock garden can be constructed as a table top Zen garden.

Fig. 11. Some children decorate rocks to personify them. Placing rocks in a pattern conveys their personal story.

(1) The concept of a Zen garden can be adapted to accept stones as the programming objects. A small motorised object can move in the white sand in response to the program instructions, leaving behind a trail in the sand. (2) The painted rocks in Fig. 11 have potential to reflect the programmer's mood and intensions. This is because the programmer himself can paint the rocks to reflect his personal interpretation of the program.

IX. Conclusion

In this paper, we reported on an ambient programming environment, the Rock Garden, where the rocks are both aesthetically appealing as well as forming a tangible program. The rocks are weather resistant because the magnets are embedded under the surface. Our programming evaluations were done with the help of pre-school children. The results show that there is no significant difference in the ability of the two groups from different socio-economic backgrounds in using the tangible programming environment. We also provided a glimpse at the potential application of this tangible programming system in the context of an artistic garden. This was done by decomposing the program into its nine steps. These steps were then "implemented" in a fictitious rock

garden. In addition, we offered two examples of crafted articles that could potentially be used as programming objects.

ACKNOWLEDGMENTS

We are grateful to Dr Foko and Mrs van Deventer who assisted in the conceptualisation, design and execution of the experiments as well as the subsequent analysis of the collected data. Mr Krause provided invaluable inputs to this paper. The authors wish to express gratitude to the Morning Star school the Zama care center in Putfontein. This research was funded by the South African Department of Science and Technology.

REFERENCES

[1] "Visual Studio Express." Microsoft, http://www.visualstudio.com/en-US/products/visual-studio-express-vs, 2014.

[2] "Delphi." Embarcadero, http://www.embarcadero.com/products/delphi, 2014.

[3] "Processing." The Processing Foundation, http://www.processing.org/, 2014.

[4] I. Harel and S. Papert, Eds., *Constructionism*. .

[5] iRobot, iRobot Corporation, http://www.irobot.com.

[6] K. P. Fishkin, "A taxonomy for and analysis of tangible interfaces," *Personal Ubiquitous Comput.*, vol. 8, no. 5, pp. 347–358, 2004.

[7] Y. Fernaeus and J. Tholander, "Finding design qualities in a tangible programming space," in *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, 2006, pp. 447–456.

[8] S. Jordà, C. F. Julià, and D. Gallardo, "Interactive surfaces and tangibles," *XRDS*, vol. 16, no. 4, pp. 21–28, 2010.

[9] M. S. Horn and R. J. K. Jacob, "Designing tangible programming languages for classroom use," in *TEI '07: Proceedings of the 1st international conference on Tangible and embedded interaction*, 2007, pp. 159–162.

[10] A. C. Smith, "Handcrafted physical syntax elements for illetterate children: initial concepts," in *Proceedings of the 7th international conference on Interaction design and children*, 2008, pp. 157–160.

[11] A. C. Smith, T. Foko, and A. Van Deventer, "Quantifying the visual perception skills of pre-school testees using a novel tangible electronic test instrument.," in *Science real and relevant: 2nd CSIR Biennial Conference*, 2008, p. 11.

[12] S. Papert, *The children's machine: rethinking school in the age of the computer*. New York, NY, USA: Basic Books, Inc., 1993.

[13] P. Markopoulos, J. C. Read, S. MacFarlane, and J. Hoysniemi, *Evaluating Children's Interactive Products: Principles and Practices for Interaction Designers (Interactive Technologies)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[14] A. C. Smith, "Visual perception skills testing: preliminary results," in *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, 2009, pp. 207–208.