Simple Tangible Language Elements for Young Children

Andrew Cyrus Smith
CSIR Meraka Institute
PO Box 395
Pretoria, 0001, South Africa
+27 12 8414626

acsmith@csir.co.za

ABSTRACT

We propose simple tangible language elements for very young children to use when constructing programmes. The equivalent Turtle Talk instructions are given for comparison. Two examples of the tangible language code are shown to illustrate alternative methods of solving a given challenge.

Categories and Subject Descriptors

D.3.2 [**Programming Languages**]: Language Classifications. D.2.6 [**Software Engineering**]: Programming Environments - *interactive environments.* H.5.2 **[Information Interfaces and Presentation**]: User Interfaces – *haptic I/O, interaction styles.*

General Terms

Design, Human Factors, Languages.

Keywords

Programming, children, syntax, tangible, turtle talk.

1. INTRODUCTION

In the late 1960's Seymour Papert and his small team developed a simple programming language aimed at children. This language is Logo [2,p210]. Papert's aim in developing the original programming syntax was to have a simple environment in which children can discover and explore the potentially exciting, creative, and stimulating world of end-user programming. To make the results of the programming concrete to the children, Papert and his team at MIT introduced a so-called turtle. The turtle existed both on the computer screen and in the real-world. Close to 40 years have passed since the original Logo development, but not much has changed in the way Logo is used: a programmer still needs to be computer literate to make full use of the programming environment.

It can be argued that a programming environment which is based on the direct manipulation of tangible programming elements can reduce the cognitive burden of the coder. In such an environment there is a direct and obvious link between the input and resultant output when coding: there is a one-to-one mapping between the tangible instruction and the resultant behaviour of the output device when the instruction is executed.

In this paper we explore an approach to simple programming which does not require either a PC or the ability to read or write (letteracy). Our approach is based on tangible programming which utilises tangible objects that are decorated with symbols that convey their meaning.

8th International Conference on Interaction Design and Children (IDC), Como, Italy, 3-5 June 2009

2. TANGIBLE PROGRAMMING

The approach relies on the direct manipulation of identical cubes. All cubes have the same symbol embedded on the surface and the symbol pertains to the immediate function of the cube. We have concretised the system inputs in the form of tangible cubes. The system output is a tangible toy car.

The system presented here abstracts [1,p82] the underlying implementation of a system that senses the specific cube orientation when placed at a discreet location, converts the electrical representation of the cube into an instruction, and sends the instruction in the form of an infra red signal to a toy car for immediate execution. The toy car itself contains an embedded processor which runs a programme for receiving infra red commands through its own sensor and controls electrical motors based on these commands. As far as the user is concerned, the cubes contain the instructions and the toy car executes them.

The system uses metaphors for the four basic actions provided. These actions are: moving a fixed distance forward and back, and turning either left or right. The embedded programme in the car is such that the car moves exactly one square forward or back, or turns 90 degrees left or right and then stops.

3. LOGO AND SYMBOLS

The geometry used is in many ways similar to Turtle geometry [2,p55]. As for the LOGO on-screen turtle, the tangible output device has both a position and a heading. Position and heading changes are effected by the cubes. Table 1 lists the symbols with corresponding TURTLE TALK commands. Figures 1 and 2 show coding examples using the cubes and TURTLE TALK, with the resultant tangible output shown as well.

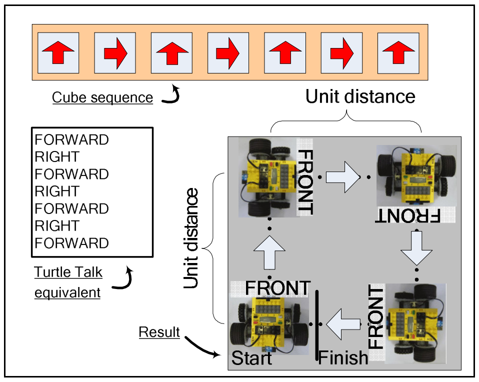
4. USAGE

Users are given a challenge to solve by making use of the four available instructions. Programming is done by placing cubes onto the programming mat in the desired orientation. The system is then activated and the toy car's motions closely observed. If the execution does not correspond to the user's intentions, the discrepancy is resolved by inspecting the sequence of cubes and comparing them with the actions the toy car executed. The cubes are then adjusted and the system activated again. This process is repeated until the toy car behaves as intended.

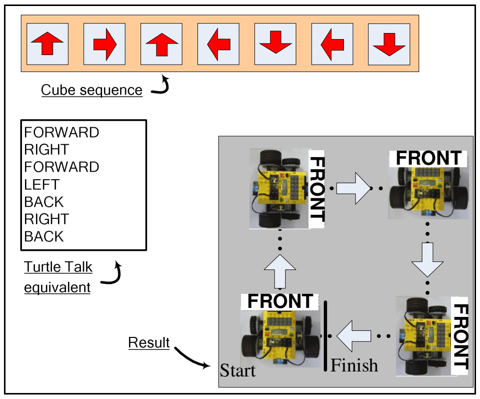
Table 1: Programming symbols and corresponding TURTLE TALK commands.

⬆ FORWARD	⬇ BACK
➡ RIGHT	⬅ LEFT

The toy car moves exactly one unit (one square) on the execution mat for each forward/reverse movement instruction received. In Figures 3 and 4, the yellow blocks indicate the recommended route to be programmed in order to meet the challenge. In one scenario the challenge is simply to get the toy car to reach both target objects #1 and #2. The second scenario requires that the toy car reverses into target object #2. Possible solutions to these challenges are given in Figure 5.



```
FORWARD
RIGHT
FORWARD
RIGHT
FORWARD
RIGHT
FORWARD
```



```
FORWARD
RIGHT
FORWARD
LEFT
BACK
RIGHT
BACK
```

Figures 1, 2: Two physical coding sequences for tracing a square using the system described. The sequence is read from left to right. TURTLE TALK code serves as a comparison. The toy car movements are shown.

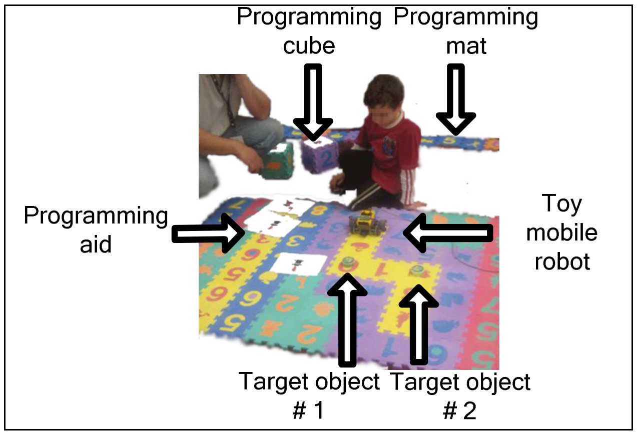


Figure 3: The test lay-out.

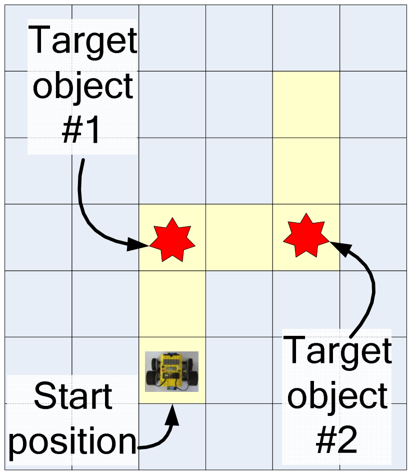


Figure 4: Diagrammatic representation of the test lay-out.

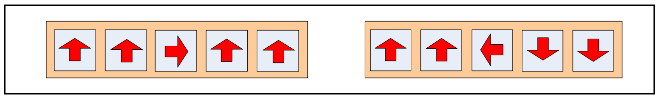


Figure 5: Solution to the first challenge (left), using forward motions only, and to the second challenge (right), including backward motions.

5. CONCLUSION

In this paper we have expressed the need for a simple tactile programming environment. We then presented an environment that potentially addresses this need. Examples of its use were given.

6. ACKNOWLEDGMENTS

This research was funded by the South African Department of Science and Technology.

7. REFERENCES

[1] Dourish, P., Where the action is, MIT Press, 2001.

[2] Papert, S. Mindstorms, Children, Computers, and Powerful Ideas, Basic Books, Inc., 1980.