Reviewing the Affordances of Tangible Programming Languages: Implications for Design and Practice

Michail N. Giannakos
Department of Computer and Information Science,
Norwegian University of Science and Technology (NTNU)
Trondheim, NO-7491 Norway
michail.giannakos@idi.ntnu.no

Abstract
During the last few years, the focus of computing education and programming in primary and secondary schools (shortly K-12) has reached a significant turning point. This study reviews the published papers on the field of tangible programming languages (TPLs) in K-12 schools in order to summarize the findings, guide future studies and give reflections for design and practice. From a systematic literature search 4 TPL peer-reviewed articles were collected and analyzed. Results of this short survey show that designers should emphasize on TPLs unambiguous manipulations, and consider clear mappings between tangible and virtual commands. Despite the challenges, the studies reviewed suggest that implementing programming lessons in K-12 education using TPL could be an enjoyable and effective learning experience.

Author Keywords
Authors' choice; of terms; separated;.

ACM Classification Keywords
K.3.1 [**Computer Uses in Education**] Computer-assisted instruction (CAI), Distance learning; H.5.3 [**Group and Organization Interfaces**]: Collaborative computing, Evaluation/methodology

Introduction

During the last few years, the focus of computing and programming education in primary and secondary schools (shortly K-12) was shifted from computer and ICT applications towards rigorous computing, programming and problem solving skills in several countries and States [5]. However, there are a number of challenges in ensuring that computing curricula, tools and environments embody appropriate progression and engender motivation for the topic across the school years

Implementing computing lessons in various ways and in a more regular basis like in the typical school environment could enhance learning and benefit students with the development of fundamental skills like computational thinking and creativity [10]. Coding could support students in many ways in a carefully designed learning setting. This perspective does not mean that all learners will become necessarily professional programmers but that they will gain useful practices and skills of the digital era.

Figure 1: Tern TPL consists of a collection of wooden blocks shaped like jigsaw puzzle pieces [4]

By involving students in the design decisions they begin to develop technological fluency and the needed for the 21st Century competences and understanding. Tangible Programing Languages (TPL) give children the opportunity to manipulate directly tangible objects which actually form the generated code. TPLs allow children to interact with physical objects and transform the logic in the physical world to the program logic. Tangible affordances have the potential to make the symbolic and abstract manipulations involved in creative procedures more concrete and manageable for young students [1].

The aim of this paper is to develop a critical discussion about the established practices on tangible programming environments (TBL), TBL affordances and expected outcomes of putting them into practice under different contexts such as school context, hackerspaces, makerspaces, FabLabs etc. This will allow us to better understand and improve the value of TBL to support teaching and learning.

Tangible Programming Languages

The Button Box [5] developed by Radia Perlman at the MIT Logo Lab in the mid-1970s is most likely the first example of tangible programming paradigm. Button Box was design to control a "floor turtle", via a Logo subset called TORTIS. Button Box's concept of procedure was too abstract for younger children. The system provided no way for a child to modify a program once it was created. Hence, if a mistake was found, the child had to recreate the entire sequence from the beginning.

For the context of this study, we have considered a peer reviewed search in the major international online bibliographic databases (AACE Digital Library, ACM DL, ERIC, Scholar etc.) [2]. This process was conducted independently by two experts, a CS educ. researcher and a research librarian and resulted in 12 articles and the following 5 different TPLs.

Tern [4]

Tern creates physical computer programs using interlocking wooden blocks (figure 1). The shape of the interlocking blocks creates physical syntax (no invalid

Figure 1: An indicative program with the TPL T_ProRob [7]

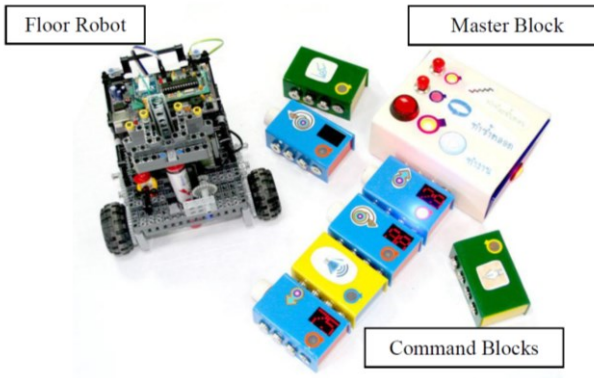

Figure 3: Robo-Blocks consists of command blocks connected to a master block which interprets the program and wirelessly sends the commands to the floor robot [8]

programs) and the tern programs can be compiled by the pressing of a button. Tern allows children to program a moving robot by putting together wooden command blocks. The program gets picked up by a camera and a computer processes the image to interpret the program.

T_ProRob [7]

The T_ProRob (tangible) system consists of 28 commands and 16 smaller parameters, all cubic shaped. Users connect the cubic commands and parameters and the program‘s execution starts by pressing a button on the top of the basis (__master box“). An indicative program structure, is shown in Fig. 1

Robo-Blocks [8]

The Robo-Blocks system controls the movement of a floor robot by physical command blocks which can be snapped together through magnetic connectors. The chain of command blocks are then attached to a master block which interprets the program sequence and transmits the commands wirelessly to the floor robot. Robo-Blocks consists of command blocks connected to a master block which interprets the program and wirelessly sends the commands to the floor robot.

T-Maze [9]

T-Maze is composed of maze game, programming wooden blocks, camera and sensor input devices (figure 4). The maze game contains two parts: maze creating tool and maze escaping game. The maze creating tool uses the tangible programming blocks to create mazes which can be used in maze escaping game directly. T-Maze needs children to control the virtual character in maze to go through relative sensor cells and finally reach the exit of the maze. The programming wooden blocks are the carriers of the TPL, which is designed for the maze game.

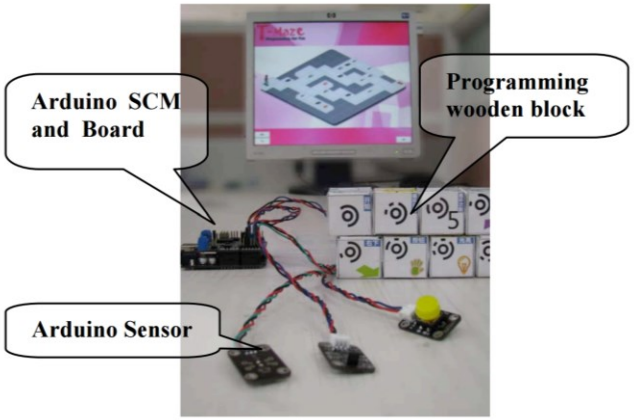

Figure 4: TPL T-Maze and Sensor Input Devices [9]

Quetzal language [3]

Quetzal is a programming language for controlling the LEGO MindstormsTM RCX brick. It consists of interlocking plastic tiles that represent flow-of-control

Figure 5: A collection of tangible programming parts from the Quetzal language [3]

structures, actions, and parameters. Statements in the language are connected together to form flow-of-control chains. Simple programs start with a Begin statement and end with a single End statement. Children can add or change parameter values to adjust the wait time and the motor's power level. The order in which the statements are connected is important, but the overall shape of a program does not change its meaning.

Implications for Design and Practice

After reviewing the 5 TPLs and the related articles, we ended up that TPLs could make programming even more attractive to children. TPLs have a great potential of helping children to better understand, locate, and solve problems. In many ways the affordances of tangible objects are less ambiguous than virtual ones, and thus more predictable to manipulate. *Designers should emphasize on TPLs ability to mimic how people interact with everyday objects, and adopt this ability to reduce children reluctance with coding*.

Another interesting insight is that, children have high difficulty on switching from TPL to more conventional IDE or even to another TPL. Hence designers should consider a clear mapping between the tangible and virtual commands. TPL has been found to reduce children thresholds for engaging with coding; however educators should develop learning concepts and scenarios aiming deeper social and personal purposes that users engage in.

Acknowledgements

The author would like to express his gratitude to C. Lawton and V. Garneli for the review search.

References

1. Cassell, J. 2004. Towards a Model of Technology and Literacy Development: Story Listening Systems, Journal of Applied Developmental Psychology 25(1), 75-105.
2. Garneli, V, Giannakos M.N., and Chorianopoulos K. 2015. Computing Education in K-12 Schools: A Review of the Literature, In Proc. of EDUCON, IEEE Press, 543–551.
3. Horn, Michael S., and Robert JK Jacob. 2007. Designing tangible programming languages for classroom use. In Proc. of TEI, ACM Press, 159-62.
4. Horn, Michael S., Erin Treacy Solovey, and Robert JK Jacob. 2008. Tangible programming and informal science learning: making TUIs work for museums. In Proc. of IDC, ACM Press, 194-201.
5. Hubwieser, P., Armoni, M., Giannakos, M.N., and Mittermeir. RT. 2014. Perspectives and visions of computer science education in primary and secondary (K-12) Schools. ACM Transactions on Computing Education (TOCE), 14 (2).
6. Pea, R. 1983. Logo Programming and Problem Solving. In Proc. of AERA conference, Montreal, Canada, April 1983.
7. Sapounidis T, and Demetriadis S. 2013. Tangible versus graphical user interfaces for robot programming: exploring cross-age children's preferences. Personal and ubiquitous computing, 17 (8), 1775-1786.
8. Sipitakiat, Arnan, and Nusarin Nusen. 2012. Robo-Blocks: designing debugging abilities in a tangible programming system for early primary school children. In Proc. of IDC, ACM Press, 98-105.
9. Wang, Danli, Cheng Zhang, and Hongan Wang. 2011. T-Maze: a tangible programming tool for children. In Proc. of IDC, ACM Press, 127-135.
10. Wing, Jeannette M. Computational thinking. 2006. Communications of the ACM 49 (3), 33-35.