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

Sofia Papavlasopoulou, Michail N. Giannakos, and Letizia Jaccheri

Department of Computer Science

Norwegian University of Science and Technology (NTNU)

Trondheim, Norway

{spapav, michailg, letizia.jaccheri} @ ntnu.no

*Abstract*— During the last few years, the development of tools for learning 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 12 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.

Keywords—Tangible programming languages; K-12 education; affordances

# 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 [1]. 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 [2]. 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. 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 [3].

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 programmming languages

The Button Box [1] 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.) [4]. This process was conducted independently by two experts, a CS education researcher and a research librarian and resulted in 17 articles and the following 12 different TPLs.

## Tern [5]

Tern creates physical computer programs using interlocking wooden blocks (Fig. 1 left, next page). The shape of the interlocking blocks creates physical syntax (no invalid programs) and the tern programs can be compiled by the pressing of a button. Blocks have different colors for different types (e.g. sound blocks, control blocks, dance moves). 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.

Tern was examined in a museum with not specified age groups; therefore, its goal is mainly that every child without previous experience in programming should be able to use it within a few minutes. The language syntax was based in a jigsaw puzzle metaphor and in systems like Scratch which minimize the possibility of a syntax error. Learning outcome was not measured in this concept but it provides possibilities to reinforce a learning process involving design, testing, reflection and revision.

## TProRob [6]

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 right, next page.

TProRob refers to ages from 5 to 12 years old. In their study Sapounidis and Demetriadis [9] focused on children’s enjoyment and easiness of the system depending on their age, and not on the learning. They showed that Tangible interface was more enjoyable and easier only for the young children. Thus, it is crucial the domains in which tangible can be beneficial.



1. Tern program using conditional branches, loops, and subroutines (left) and an indicative program with T\_ProRob (right) [5][6].

## Robo-Blocks [7]

The Robo-Blocks system controls the movement of a floor robot by physical command blocks which can be snapped together through magnetic connectors (Fig. 2 up). The chain of command blocks is then attached to a master block which interprets the program sequence and transmits the commands wirelessly to the floor robot.

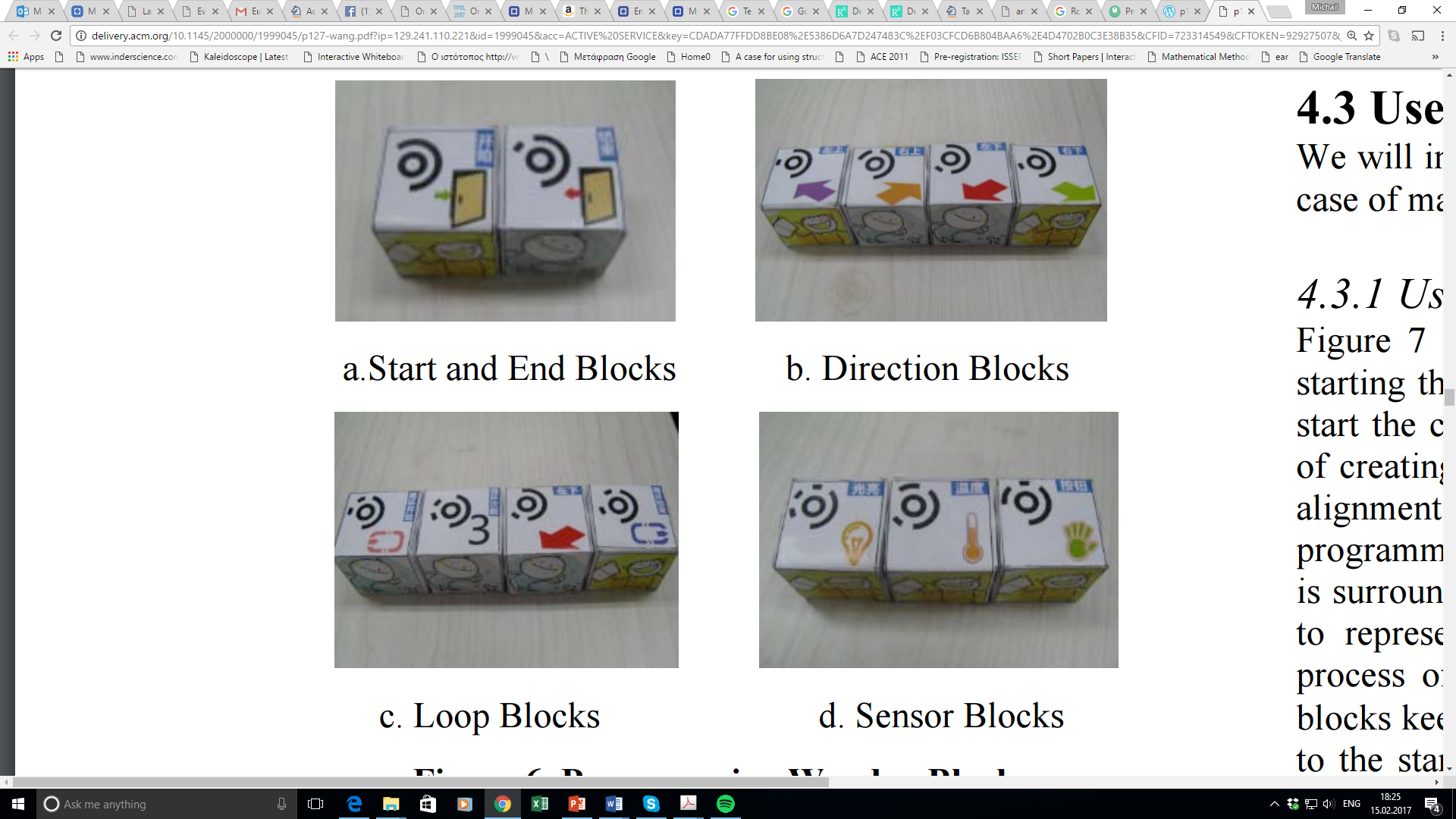
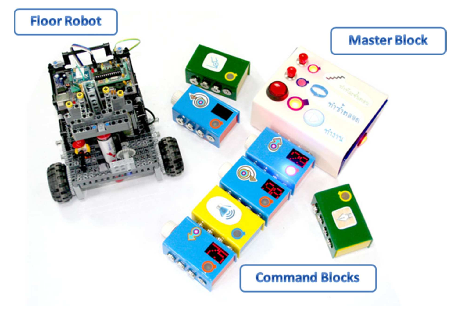
Using Robo-Blocks children ages 5 to 12 years old can increase their ability in debugging. During the process, children run their program and gradually use passive objects, first to identify and then solve these problems. Older students can manage this process more effectively than the younger ones.

## T-Maze [8] and E-Block [9]

T-Maze is composed of maze game, programming wooden blocks, camera and sensor input devices (Fig. 2 down). 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. There are four kinds of programming wooden blocks: start and end block, direction block, loop block, and sensor block. The programming wooden blocks are the carriers of the TPL, which is designed for the maze game.

E-Block was designed to save the problems of the T-Maze system. It consists of four parts: the maze game, the programming blocks, the wireless box and the sensors. Blocks of E-Block system have also LED as feedback. The main difference from T-Maze is that E-Block has SCM (simple chip microcomputer) that enables children to place the blocks without thinking he camera’s view scope, as they did in the camera version tool, T-Maze.

The target group of both T-Maze and E-Block is from 5 to 9 years old. T-Maze focuses on enhancing children’s logical thinking abilities and problem solving skills. A real-time camera captures their programming and gives instant feedback, making the programming process easier. However, T-Maze has limited types of sensors, programming blocks, and games. On the contrary, E-Block provides a higher proportion of programming operations and lower space related expressions.

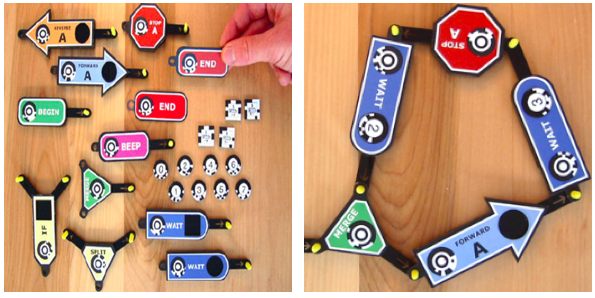


1. Robo-Blocks System (up) and the T-Maze Programming Wooden Blocks (down) [7] [8] [ 9].

## Quetzak Language [10]

Quetzal is a programming language for controlling the LEGO MindstormsTM RCX brick (Fig. 3, next page). It consists of interlocking plastic tiles that represent flow-of-control structures, actions, and parameters. Statements in the language are connected 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.

This language consists of parts with no embedded electronics or power supplies and use portable scanning and inexpensive parts that make it suitable for classroom use. The goal is that the children can understand the effects of their programs on the robots and even predict and identify bugs in their code, writing more sophisticated programs.



1. Some of the tangible programming parts from the Quetzal programming language (left) and an example of a created loop [10].

## CHERP [11]

CHERP (Creative Hybrid Environment for Robotic Programming) is a hybrid tangible/graphical programming language for young children with which you can create programs for robots like LEGO Mindstorms, Lego WeDo and the KIWI research prototype[[1]](#footnote-1)1. Using CHERP children can create physical programs with interlocking blocks (wooden cubes or just printed icons provided for tangible interaction) (Fig. 4) or graphical programs onscreen or even a combination of them. Regarding the physical blocks, there are no embedded electronics or power supplies, instead a standard webcam is needed to take o picture of the the program and a circular bar-code like TopCodes will convert the blocks into digital code. The basic CHERP syntax contains begin and end blocks, control flow blocks (IF, IF NOT, REPEAT). It is also possible to build your own robot using LEGOTM bricks, robotic parts and recycled materials.

CHERP focuses on teaching kindergarden students the relationship between the blocks and the movements of their robots. In general, through their interaction with the CHERP programming language, children can learn basic problem solving and computational thinking skills, as well as social-emotional development.



1. Different blocks of CHERP program (left) and an example with their representation on wooden blocks (right) [11].

## LittleBits [12]

LittleBits are discrete modular electronic components that can be assembled magnetically to form in tiny circuits and interactive devices. There are 60 different modules working with millions of combinations. Each of the modules are outputs that send signals to the circuit. For example, green modules are outputs that add motion, light; orange modules are wires helping to extend the circuits; pink help interactions with buttons, dimmers, sensors and more (Fig. 5). LittleBits have a variety of versions according to different uses (home kit, workshop set, STEAM student set). Arduino coding kit is a tangible programming language that combines the power of Arduino and the magnetic components without soldering and wiring. It provides an introduction to programming by learning the basics of coding while creating and program specific inventions or upload code from the site.

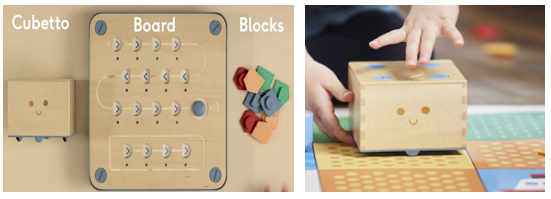
LittleBits support all STEAM subjects providing a range of possibilities through their variety of different kits, supporting specific needs, starting from the age of 8. Their main characteristic is the instant feedback that permits fast iterations and a constructive interaction. The relation between action, reaction and functionality supports a creative learning environment of computational thinking and problem solving skills.



1. LittleBits different components (left) and an example of a built robot (left) [12].

## Cubetto [13]

Cubetto is a tangible programming language for children with a starting age of 3 years-old. Cubetto consists of a wooden play set and focuses on teaching toddlers and preschool children the main principles of coding without using a computer. The set includes a wooden cube-shaped robot maned “Cubetto”, a programming board and many different instruction blocks with colors (Fig. 6, next page). Different commands have different colored blocks, for example yellow is to turn right and green is a function key. “Cubetto” moves according to the directions given in the programming board with the use of open-source electronics platform Arduino. Each block has an action and can be combined to create programs, however it provides only the basic programming concepts.

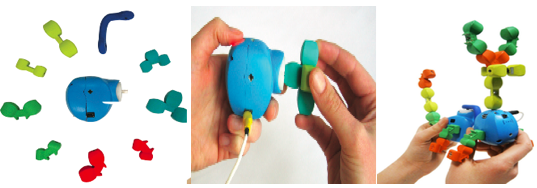


1. Cubetto robot, the control board and the coding blocks (left), the cubetto robot in motion on the specifically designed maps (right) [13].

## Topobo [14]

Topobo is a 3D tangible construction kit that can record, playback and have a physical motion. Its components can be assembled quickly in different shapes, like animals. These components are Passive (static), Active (motorized), basic Queens, power suppliers and cables, by using programming by demonstration model a creature can be programmed to move in a specific way, push, pull and twist, or just simple make them walk for 30cm (Fig. 7). Each built creature can “learn” a specific movement which will be repeatedly played unless a special active “Queen” is used that forces all the other actives to mimic its motion. Special booklets give the instructions for programming, describe the concept, design and technical details.

Topobo, can support scientific learning by showing the physical connection of math and science ideas and giving a tangible representation of a dynamic behaviour. Children as young as 4 years old can learn about movement and animal locomotion an explore physics concepts like dynamic balance, centre of gravity leverage and system behaviour.

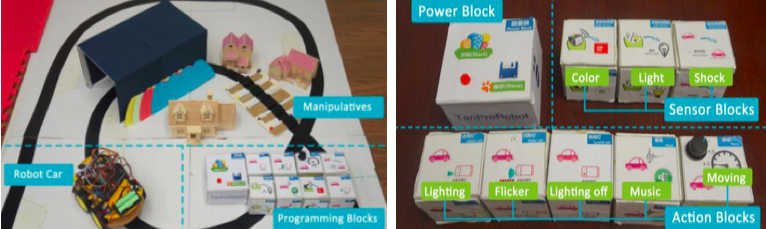


1. The Topobo system (left), programming Topobo (middle) and an example of a designed Tobopo “animal” (right) [14].

## TanProRobot [15, 16]

Tan ProRobot 2.0 is a tangible system originally designed for children grades 1-2 to learn programming concepts. The system consists of three parts: tangible blocks, a robot car (the output device) and manipulatives (Fig. 8). At the first stage, children give actions to the car by arranging the programming blocks (sensor block, action block and power block) and then create a road for the car to move, decorating it with manipulatives. Interactive manipulatives (e.x. color papers, a tunnel) are recognized by the robot car’s sensors and it reacts accordingly. Children can test their programs each time they complete an event handling program fragment, and change the game scene according to their will.

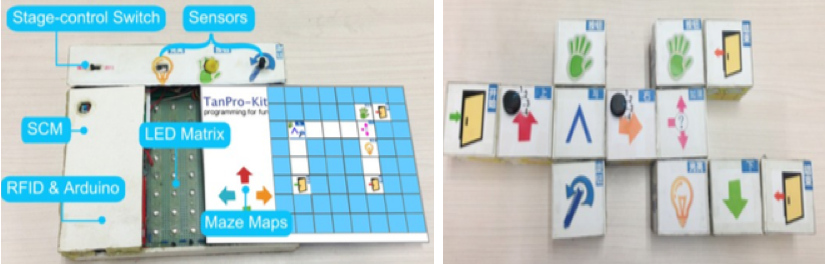
Tan ProRobot is a tool easy to use helping children understand core computer science concepts and introduce them in event handling concepts. However, the system does not include a range of sensors and computing abilities.



1. Topobo system and its components (robot car, manipulatives, programming blocks) (left) and the different programming blocks (right) [15].

Tanpro-Kit 2.0 is a tangible programming tool for children, composed by: (a) LED Pad, (b) Maze Maps, (c) Start /End Block, (d) Direction Blocks, (e) Direction blocks with parameters, (f) Sensor Blocks, and (g) Logic Blocks & Branch Block. The game has two main stages, the programming stage and the running stage (Fig. 9). Children use the programming blocks to create a routing path while the LED pad provides them instant feedback confirming a correct placement of a block. The LED pad will execute the different command sequences.

Tanpro-Kit aims to teach three important programming concepts: parameters, Boolean logic and branch. Different logic blocks introduce the related concepts. For example, the AND block and OR block are designed to introduce Boolean Logic. Further improvement is needed to extend programming concepts and improve their representation for better understanding.



1. The LED pad of the Tanpro-Kit and its different parts (left) and the programming blocks in a sequence (right) [16].

# Implications for design and practice

After reviewing the 12 TPLs and the related articles, (see in the appendix a table with the characteristics of the TPLs) we ended up that TPLs could make programming even more attractive to children of any age, starting from 3 years-old. 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. TPLs apparently offer the entertaining aspect of toys/games and at the same time offer possibilities of an instructional tool. Designers should emphasize on TPLs ability to mimic how people interact with everyday objects, and adopt this ability to reduce children reluctance with coding. The common characteristic of all TPL is their easiness and simplicity. Due to the nature of TPL, the programming process requires the alignment of the mental concept with the tangible shape. Also, TPL can be more effective than simple programming languages for early stages of programming in terms of convenience and understanding, but they are similar in terms of enjoyment [17]. TPLs have been successfully applied in different contexts, from classrooms to museums. The studies evaluated reveal that TPLs have been used from the age of 3, but some had different results depending on age and gender. Younger children find the usage of the TPLs easier than the old ones who find them more enjoyable [6]. Comparing TPLs to graphical systems, older children who have more experience with computers tend to be attracted from graphical environments.

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. All teachers need to acquire knowledge to support the computational aspects of tangible programming languages. One of the challenges teachers face is the enhancement of their ability to improvise, since the computational aspect of tangibles is not similar to standard teaching activities. Moreover, a combination of different TPLs provide opportunities for further development considering different purposes [6]. In order to have an effective learning environment, a TPL needs to allow for iterative explorations of concepts. Therefore, it is important to have a predefined learning goal [16] [15]. The use of TPLs may be beneficial for different ages, besides, focusing on a specific target age, should help to design more effective and efficient TPLs. Also, there is a need for formal evaluations and more implementations in school classrooms or after school programs.

##### Acknowledgment

This work was funded by the Norwegian Research Council under the projects FUTURE LEARNING (number: 255129/H20), the Centre for Excellent IT Education (ExcITEd - <http://www.ntnu.edu/excited>) and by the GEMINI Center in Technology Enhanced Lifelong Learning (<http://www.tellgemini.org/>).

##### References

[1] P. Hubwieser, M. Armoni, M.N. Giannakos, R.T. Mittermeir, Perspectives and visions of computer science education in primary and secondary (K-12) Schools, *ACM Transactions on Computing Education (TOCE)*, **14** (2014) 7.

[2] J.M. Wing, Computational thinking, *Communications of the ACM*, **49** (2006) 33-35.

[3] J. Cassell, Towards a model of technology and literacy development: Story listening systems, *Journal of Applied Developmental Psychology*, **25** (2004) 75-105.

[4] V. Garneli, M.N. Giannakos, K. Chorianopoulos, Computing education in K-12 schools: A review of the literature, in: Global Engineering Education Conference (EDUCON), 2015 IEEE, IEEE, 2015, pp. 543-551.

[5] M.S. Horn, E.T. Solovey, R.J. Jacob, Tangible programming and informal science learning: making TUIs work for museums, in: Proceedings of the 7th international conference on Interaction design and children, ACM, 2008, pp. 194-201.

[6] T. Sapounidis, S. Demetriadis, Tangible versus graphical user interfaces for robot programming: exploring cross-age children’s preferences, *Personal and ubiquitous computing*, **17** (2013) 1775-1786.

[7] A. Sipitakiat, N. Nusen, Robo-Blocks: designing debugging abilities in a tangible programming system for early primary school children, in: Proceedings of the 11th International Conference on Interaction Design and Children, ACM, 2012, pp. 98-105.

[8] D. Wang, C. Zhang, H. Wang, T-Maze: a tangible programming tool for children, in: Proceedings of the 10th International Conference on Interaction Design and Children, ACM, 2011, pp. 127-135.

[9] D. Wang, Y. Zhang, S. Chen, E-Block: A tangible programming tool with graphical blocks, *Mathematical Problems in Engineering*, **2013** (2013).

[10] M.S. Horn, R.J. Jacob, Designing tangible programming languages for classroom use, in: Proceedings of the 1st international conference on Tangible and embedded interaction, ACM, 2007, pp. 159-162.

[11] A. Strawhacker, A. Sullivan, M.U. Bers, TUI, GUI, HUI: is a bimodal interface truly worth the sum of its parts?, in: Proceedings of the 12th International Conference on Interaction Design and Children, ACM, 2013, pp. 309-312.

[12] A. Bdeir, Electronics as material: littleBits, in: Proceedings of the 3rd International Conference on Tangible and Embedded Interaction, ACM, 2009, pp. 397-400.

[13] Primotoys, https://www.primotoys.com/ retrieved on 15.02.2017.

[14] A.J. Parkes, H.S. Raffle, H. Ishii, Topobo in the wild: longitudinal evaluations of educators appropriating a tangible interface, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2008, pp. 1129-1138.

[15] D. Wang, L. Zhang, C. Xu, H. Hu, Y. Qi, A Tangible Embedded Programming System to Convey Event-Handling Concept, in: Proceedings of the TEI'16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction, ACM, 2016, pp. 133-140.

[16] D. Wang, L. Zhang, Y. Qi, F. Sun, A tui-based programming tool for children, in: Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, ACM, 2015, pp. 219-224.

[17] D.-Y. Kwon, H.-S. Kim, J.-K. Shim, W.-G. Lee, Algorithmic bricks: a tangible robot programming tool for elementary school students, *IEEE Transactions on Education*, **55** (2012) 474-479.

Appendix:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TPL:** | **Program:** | **Material:** | **Own robot:** | **Connected to different types of robots:** | **Definition of each instruction:** | **Shape:** |
| Tern [5] | Program converter from a picture | Wooden blocks | Yes | Robot on a display | Colors | Cubes (like puzzle pieces) |
| T\_ProRob [6] | Program is sent to a remote computer and then to the robot which starts by pressing the  “master box” with a button to start the robot | Plastic blocks |  | Lego NXT robot | Shapes and colors | cubes |
| Robo-Blocks [7] | “master block” to interpret the programs and transmits the commands wirelessly to the floor robot | Magnetic command blocks | Yes |  | Colors | Parallelogram |
| T-Maze [8] | Arduino | Wooden blocks | Yes | control the virtual character | Figures | Cube |
| E-blocks [9] | Single chip microcomputer | Wooden blocks | Yes | control the virtual character | Figures | Cube |
| Quetzal Language [10] | Program converter from a picture | Plastic tiles | - | Lego Mindstorms TM RCX | Shapes/colors | Variety of shapes |
| CHERP [11] | Connection with the robot | - | - | Many types of robots | Figures | Square |
| Littlebits  (coding kit) [12] | Program converter from a picture | * Wooden * Create also your own | - | Lego we do, Lego Mindstorms, KIWI | Figures/colors | Cube |
| Cubetto [13] | Arduino | Magnetic components | - | - | Colors/shapes | - |
| Topobo [6] | Arduino | Wooden blocks | Yes | - | Colors | Cube |
| TanProRobot [15] | Single chip microcomputer/Arduino | Plastic/wood | Yes | - | Colors/Shapes | Variety of shapes |
| TanPro-Kit [16] | Single chip microcomputer/Arduino | Plastic/wood | No | - | Colors/Shapes | Cube |

1. 1 http://ase.tufts.edu/DevTech/ReadyForRobotics/index.asp [↑](#footnote-ref-1)