Block-Oriented Programming with Tangibles: An engaging way to learn Computational Thinking Skills

Alessio Malizia
School of Creative Arts., University of Hertfordshire, UK
alessio.malizia@oldsport.org

Tommaso Turchi
Computer Science Dept., Brunel University London, UK
Tommaso.Turchi@brunel.ac.uk

Kai A. Olsen
Molde University College and Dept. of Informatics,
University of Bergen, Norway
kai.olsen@himolde.no

***Abstract*—Block-oriented programming environments, such as Scratch and AppInventor, have become mainstream and have helped introduce non-programmers to algorithmic thinking; that is, to computational thinking (CT) skills. In various experiments, referenced in this paper, we observed how objects (tangibles) augmented with digital properties – that is, tangible user interface (TUI) objects – can help develop CT skills. We investigate which paradigm can be used to introduce CT skills with TUIs. By drawing on past experience, we conclude that block-oriented programming provides a suitable answer to our research questions. Furthermore, we introduce a framework for implementing block-oriented programmable objects that we believe can help end-users learn CT skills.**

Keywords—Block-Oriented Programming, Computational Thinking Skills, Tangible User Interfaces.

I. INTRODUCTION

Software surrounds every aspect of life in developed countries, which implies that computational literacy and coding are desired skills for an increasingly wide audience. Many emerging initiatives, such as the Hour of Code [1], have been successful at introducing programming to a wide variety of people with little or no programming experience. These initiatives are not only about programming, but also about which skills are needed to express problem solving in a computational form. This form of algorithmic thinking is called computational thinking (CT).

Wing [2] first defined CT as a set of thinking skills, habits, and approaches that are needed in order to solve complex problems using a computer. These skills are widely applicable in our information society. CT covers far more than programming itself; it also includes a range of mental tools that reflect fundamental principles and concepts of computer science, such as abstracting and decomposing a problem, identifying recurring patterns, and being able to generalize solutions. Learning CT skills can be an effective way of introducing lay users to programming by gradually leading them to algorithmic thinking. It can be quite challenging as well. For instance, manipulating highly abstract concepts can often be a challenge for novice users. Good training and extended practice are necessary before users can master these concepts.

We argue that exploiting our innate dexterity for manipulating objects in the physical world could be an effective way to help users grasp abstract concepts that involve coding, thus developing their CT skills. Physical manipulation sits at the core of an emerging digital interaction paradigm designed with the aim of providing users with an easy-to-use interface [3]. There is a requirement that this interface can be used even by inexperienced people. This paradigm is known as tangible user interfaces (TUIs) [4,5].

Researchers in this area are also debating how TUIs reflects on learning [6, 7, 8], with specific reference to highly abstract concepts. This stems from Piagetian theories supporting the development of thinking – particularly in young children – through manipulation of concrete physical objects. A famous study [9] found that most first year students in physics courses are still in the concrete operational phase, and are therefore incapable of grasping abstract concepts not firmly embedded in their concrete experience. Other studies [10, 11] have suggested mitigating this effect by developing CT skills. It is therefore reasonable to expect that tangibles can aid in facing abstractions and learning CT skills. The question is then: *how can tangible user interfaces be applied in order to introduce algorithmic thinking and computational thinking skills to novices?*

II. BLOCK-ORIENTED PROGRAMMING WITH TANGIBLES

We believe that exploiting the benefits of a tangible interaction in conjunction with block-oriented programming, leveraging our natural ability to manipulate objects in the real world, can help learners grasp highly abstract concepts; that is, develop their CT skills. Indeed, block programming environments seem to provide a natural paradigm to model problem solving with tangibles by linking physical and digital properties to algorithmic thinking.

As demonstrated by the recently arisen blocks programming environments' research community [12], block-oriented programming presents program logic (algorithmic thinking in CT) as compositions of visual blocks. Tools such as Scratch, Blockly, Code.org's lessons, and App Inventor have introduced programming and computational thinking to a huge audience, reaching people of all ages and backgrounds. Our hypothesis, then is that: *Block-oriented programming is the paradigm to use*

in order to support the development of computational thinking skills using tangibles.

To test our hypothesis, we organized different events with audiences of non-programmers in an attempt to understand which paradigm they would envisage as useful to effectively solve problems (using or learning CT skills) with tangible user interfaces [13, 14, 15].

For instance, in one of our studies [13] we gathered five experts with background in different design areas for a one-hour focus group: three experienced interaction designers with some basic programming knowledge – one with a specific background on information visualization and one with substantial industry experience – and two product designers with no programming experience at all. We showed them some examples of workflow programming using IFTTT (IF This Then That), a widely popular Web mashup system. IFTTT allows users to create simple event-based if-then-style workflows with different Web services and acts as a hub connecting their events' triggers with actions. IFTTT was used as an example of algorithmic thinking. We successively showed them a video about an existing TUI system – the Tangible 3D Tabletop [16] – that summarized the benefits of this interaction paradigm; in particular, we highlighted the different metaphors involved in tangible systems in relation to the physical and the digital domain.

During the workshop, the participants discussed the digital appearance of the tangible elements and proposed a match between the shapes of tangibles and their digital representations. From the focus group we ran during the workshop, it emerged that the digital representation of tangible elements could help users understand the constraints on various components. This could be achieved by different and similar colors or shapes for incompatible and compatible components.

In another study involving undergraduate students [14], we interviewed three groups, with four, five, and six members, respectively, of second-year computer science undergraduate students. Each group developed an Android application as part of their annual curriculum under the supervision of a teaching staff member. They meet at least once a week to discuss progress and to work collaboratively on their next development task. Students could choose the most suitable tools to work in a group such as e-mail or collaborative tools like Trello or Slack but they would normally meet every week face to face. We let them freely explore an interactive tabletop with tangible interaction. Their smartphones were used as tangible objects that could interact with digital contents on the tabletop. We then carried out a semi-structured interview about potential scenarios.

From the interviews we observed that students wanted to employ digital services to conduct their weekly meeting. For example, to display a document on the tabletop where they could take notes, distribute the notes to others, download documents from the digital library, and perform project planning activities such as assigning different members to different roles and features in developing their android app. All these activities were basically about using tangibles to compose a workflow made of services that could be communicated to all group members: uploading a document from Dropbox, discussing and annotating the document, re-distributing the document on a group's mailing list, etc.

Starting from scenarios proposed by students, we developed a TAngible Programmable Augmented Surface (TAPAS) [13]. TAPAS is based on the premises of block-oriented programming environments such as Scratch. The idea is to support the development of computational thinking with low floor and high ceiling – i.e. enabling any beginner to cross the threshold to create working programs easily (low floor), but at the same time satisfying the needs of more advanced users (high ceiling). Moreover, supporting Computational Thinking development stems also from providing the right challenges to get from the "floor" to the "ceiling", i.e. scaffolding [17].

In TAPAS (Figure 1), users can develop simple workflows by assembling different services by means of a puzzle. Each service is mapped to a puzzle piece and its shape dictates constraints on its required inputs and outputs. Even though components are not physically represented here, but only virtually (that is, projected over the surface), the system is controlled through a tangible object. This could be a smartphone, but the system can work even with other types of objects. That is, even if the representation is virtual, the system is used as if the objects were physical, making it fun and easy to use. The digital representation of the programming constructs involved makes it a rather flexible environment that can be easily repurposed to fit many heterogeneous contexts, such as block-oriented programming.

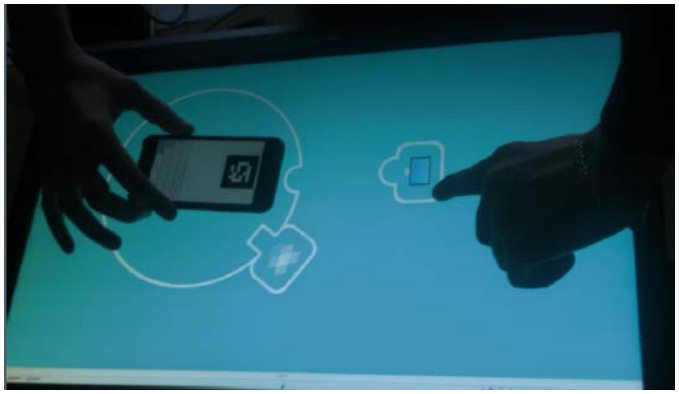

Figure 1. Block-oriented programming with tangibles – the TAPAS system

III. A FRAMEWORK FOR IMPLEMENTING BLOCK-ORIENTED PROGRAMMABLE OBJECTS

From the experiences gathered in previous studies and by evaluating end users' reactions to TAPAS, we figured out that a framework was needed to experiment with block-oriented programmable objects as a way of introducing computational thinking skills. This framework should enable non-programmers to easily evolve towards algorithmic thinking and eventually programming.

In previous works, different tangible programming environments have been explored. For example, Qi et al. [18] introduced TanProStory, a tangible programming tool to convey object-oriented programming to children using storytelling. The Digital Dream Lab [19] focus on Tangible interaction linking digital data with physical forms to support embodied use. It aims at introducing a simpler range of concepts relating to the clustering and manipulation of data to young children (5 and

under). In Blinky Blocks [20], Kirby et al., present a set of physically distributed units and introduce the use of the distributed programming language Meld to program ensembles of these units; the showed how to create 100 of such modules to provide tangible distributed programming.

Differently from the examples above, in our work we were inspired by the tangible user interfaces objects (TUIO) protocol. The TUIO protocol, as Kaltenbrunner et al. [21] stated, "is an attempt to provide a general and versatile communication interface between tangible tabletop controller interfaces and underlying application layers. It was designed to meet the needs of tabletop interactive multi-touch surfaces, where the user is able to manipulate a set of objects and draw gestures onto the table surface with the fingertips."

We designed a tangible user interface block-oriented programmable objects (TUIBOPO) framework (Figure 2), an extension of the TUIO protocol that provides further interaction capabilities for multi-device environments. This allowed us to experiment further with TAPAS and block-oriented programmable objects.

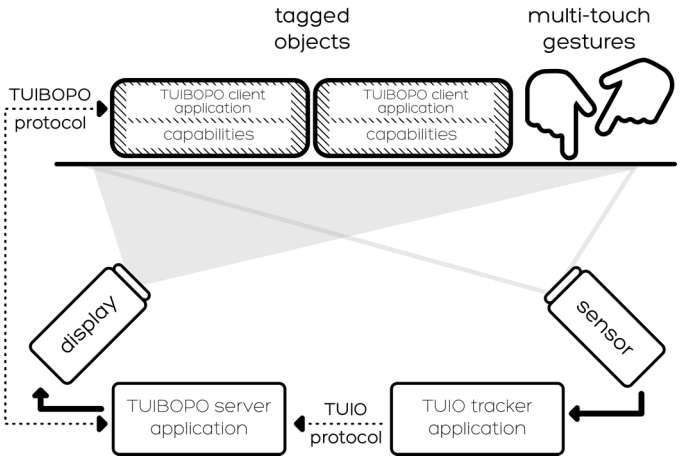

Figure 2. TUIBOPO Framework Architecture.

TUIBOPO is built on TUIO and extended to provide an abstraction layer over the capabilities of the tagged smart objects that are handled by TUIO. Our aim was to encapsulate the capabilities of a smart object – namely the properties that the physical object offers to the environment and that can be controlled and detected remotely, doing this within its virtual TUIO representation. This is a corresponding shape indicating the functionalities of the digital block (that is, rounded shaped connector for an output block as shown by the display icon in Figure 1), that enables developers to fully exploit the features of objects, such as the inputs and outputs channels that the objects might provide, either visual or tactile (in the form of a display or a physical button).

Supported objects' capabilities can be:

- Interaction capabilities; that is, buttons, multi-touch events, mid-air gestures.
- Display capabilities; that is, LEDs, screens.
- Retrieval capabilities; that is, access to storage, user's details (such as his/her Facebook account).
- Affordance capabilities; that is, shape, haptic.

The TUIBOPO environment comprises of (1) a sensor used by the TUIO component to track tagged objects and multi-touch gestures happening over the tabletop surface, whose positions are transmitted to (2) a display running the TUIBOPO server application. The server application implements a TUIO client and stores each object's movements and smart capabilities together, controlling the display output of the tabletop and the inputs and outputs of every object through the TUIBOPO protocol.

TUIBOPO can be considered a framework for implementing block-oriented programmable objects that simplifies the implementation of such objects with physical and digital properties. The idea is to help end users learn CT skills through block-oriented programmable objects.

DISCUSSION AND FURTHER WORK

Based on the idea that CT abilities such as abstraction can be developed through manipulation of concrete physical objects, we investigated which paradigm might be most suitable in order to introduce CT skills with tangibles. Drawing from literature and personal experience, we formulated the hypothesis that block-oriented programming is a suitable paradigm to use when employing TUIs to help non-programmers develop algorithmic thinking.

One of the challenges of implementing block-oriented programming with tangible objects is the double nature of the blocks (objects): physical – with an affordance communicating a specific meaning to a user – and the virtual properties of the objects. For instance, a block molded physically as a puzzle piece might map digitally to an input statement (digital representation), requiring some data that can be obtained from a further block, shaped so that it can match the input block and digitally representing a constant value. The same properties of block-oriented programming are inherited by the physical blocks (for instance, matching shapes to reduce typing errors), but generating such forms of tangible interaction might be quite complex due to the need to assemble an ecology of devices.

Using blocks allows users to be guided when dealing with different functions and their constraints – either physical or digital – and can be exploited as a way of fostering CT skills. For example, a user needing to mashup data onto a map of the closest museums to his current location can proceed as follows in a TUIBOPO-enabled system: first he selects the block that retrieves the user's location using an object supporting the location capability (e.g. a smartphone with GPS), and then pass it along to another block that produces a map of restaurants close to a given location; the output shape of the former and the input shape of the latter will have to be the same, guiding the user in composing the right set of actions needed to accomplish his goal. Users can only select available blocks with an input shape corresponding to the latest output shape they chose using an object that supports its required capabilities; the process aids users in achieving their goals and fostering their algorithmic thinking.

To help supporting such scenarios we introduced the TUIBOPO framework for implementing block-oriented programmable objects; this framework simplifies the implementation of objects with physical and digital properties.

We expect that this will aid end-users in learning CT skills. In the future, we plan to experiment with different implementations of block-oriented programmable objects in TUIBOPO. Our aim is for the digital and physical components to seamlessly interact and thus provide an environment to create a customized domain-specific block-oriented programmable objects environment.

REFERENCES

[1] https://hourofcode.com

[2] J. M. Wing, "Computational thinking," Commun. ACM, vol. 49, no. 3, p. 33, Mar. 2006.

[3] A. Bellucci, A. Malizia, P. Díaz, and I. Aedo, "Don't touch me," New York, New York, USA, 2010, p. 391.

[4] [H. Ishii and B. Ullmer, "Tangible bits," presented at the the SIGCHI Conference, New York, New York, USA, 1997, pp. 234–241.

[5] T. Turchi, A. Malizia, "A Human-Centred Tangible approach to learning Computational Thinking". ICST Trans. Ambient Systems 3(9): e6 (2016).

[6] M. Horn, E. T. Solovey, J. Crouser, R. Jacob, "Comparing the use of tangible and graphical programming languages for informal science education", in: the SIGCHI Conference, ACM Press, New York, New York, USA, 2009, p. 975.

[7] P. Marshall, "Do tangible interfaces enhance learning?", in: the 1st international conference on Tangible and embedded interaction, ACM Press, New York, New York, USA, 2007, p. 163–170.

[8] A. N. Antle, A. F. Wise, "Getting down to details: Using theories of cognition and learning to inform tangible user interface design", Interact Computing (2013) 25 (1): 1–20, 2013.

[9] K. A. Williams and A. Cavallo, Reasoning Ability, Meaningful Learning, and Students' Understanding of Physics Concepts. Journal of College Science Teaching, 1995.

[10] D. Wang, T. Wang, Z. Liu, A, "Tangible Programming Tool for Children to Cultivate Computational Thinking", The Scientific World Journal 2014(3):428080 · February 2014.

[11] M. Horn, J. Crouser, M. U. Bers, "Tangible interaction and learning: the case for a hybrid approach", Personal and Ubiquitous Computing, 16 (2012) 379–389.

[12] "Foreword," presented at the Blocks and Beyond Workshop (Blocks and Beyond), 2015 IEEE, 2015.

[13] T. Turchi, A. Malizia, A. Dix, "TAPAS: A tangible End-User Development tool supporting the repurposing of Pervasive Displays", Journal of Visual Languages and Computing, Available online 8 December 2016, ISSN 1045-926X.

[14] T. Turchi, A. Malizia, "Pervasive Displays in the wild: employing End User Programming in adaption and re-purposing", In Proceedings of the international conference on End-user development (IS-EUD'15), LNCS Vol. 9083, pp.223–229, Springer, 2015.

[15] T.Turchi, A. Malizia, "Fostering Computational Thinking skills with a Tangible Blocks Programming Environment", In Proceedings of the 2016 IEEE Symposium on Visual Languages and Human-Centric Computing (Vl/Hcc 2016).

[16] P. Dalsgaard, K. Halskov, Tangible 3D tabletops, interactions 21, 42–47, 2014.

[17] A., Repenning, D., Webb and A. Ioannidou, "Scalable game design and the development of a checklist for getting computational thinking into public schools". In the 41st ACM technical symposium (New York, New York, USA: ACM Press): 265–269, 2010.

[18] Y. Qi, D. Wang, L. Zhang, and Y. Shi, "TanProStory: A Tangible Programming System for Children's Storytelling". In Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '15). ACM, New York, NY, USA, 1001-1006, 2015.

[19] H. Oh, A. Deshmane, F. Li, J. Y. Han, M. Stewart, M. Tsai, X. Xu, and I. Oakley, "The digital dream lab: tabletop puzzle blocks for exploring programmatic concepts". In Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction (TEI '13). ACM, New York, NY, USA, 51-56, 2013.

[20] B. T. Kirby, M. Ashley-Rollman, and S. C. Goldstein. Blinky blocks: a physical ensemble programming platform. In CHI '11 Extended Abstracts on Human Factors in Computing Systems (CHI EA '11). ACM, New York, NY, USA, 1111-1116, 2011.

[21] M., Kaltenbrunner, t., Bovermann, R., Bencina, R. and E. Costanza, "Tuio: A protocol for table-top tangible user interfaces". In Proceedings of the The 6th International Workshop on Gesture in Human-Computer Interaction and Simulation, Vannes, France, 2015.