Material Programming: a New Interaction Design Practice

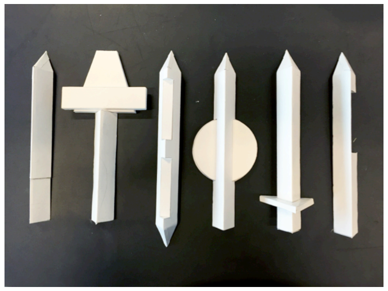

Figure 1 Speculative tools for material programming.

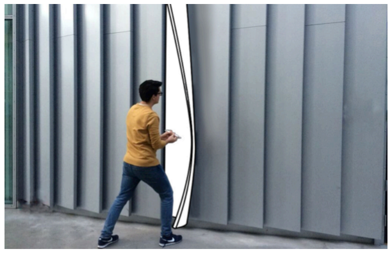

Figure 2 Illustration of imagined embodied material programming practice. Here the wall is programmed into a shape-changing behavior around the hand-held tool.

Anna Vallgårda
IxD lab
IT University of Copenhagen
Copenhagen, Denmark
akav@itu.dk

Laurens Boer
IxD lab
IT University of Copenhagen
Copenhagen, Denmark
laub@itu.dk

Vasiliki Tsaknaki
Mobile Life @ KTH
Royal Institute of Technology
Stockholm, Sweden
tsaknaki@kth.se

Dag Svanæs
IxD lab
IT University of Copenhagen
Copenhagen, Denmark
&
Department of Computer and Information Science
Norwegian University of Science and Technology.
Trondheim, Norway
dag.svanes@idi.ntnu.no

DIS '16 Companion, June 04-08, 2016, Brisbane, QLD, Australia
ACM 978-1-4503-4315-2/16/06.
http://dx.doi.org/10.1145/2908805.2909411

Abstract

We propose the notion of material programming as a new practice for designing future interactive artifacts. Material programming would be a way for the interaction designer to better explore the dynamics of the materials at hand and through that familiarity be able to compose more sophisticated and complex temporal forms in their designs. As such it would blur the boundaries between programming and crafting these new smart and computational materials. We envision a material programming practice developed around physical tools (e.g. Fig 1) that draw on bodily skills and experiences (Fig 2) while enabling actions performed directly on the material with immediate effects (no program vs. execution mode). Finally, the tools would enable one layer of abstraction and as such encompass the potential of the computational materials but not that of possibly adjacent computers, which could run more complex algorithms.

Author Keywords

Material programming; computational composites; materials; design practice; programming practice

ACM Classification Keywords

H.5.m. Information interfaces and presentation: Miscellaneous

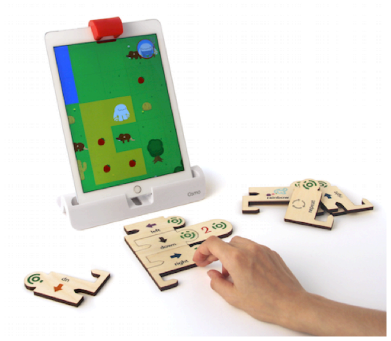
Figure 3 Tangible programming: Strawbies [5].

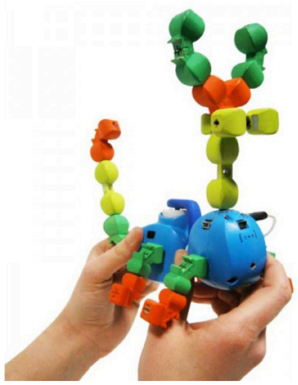
Figure 4 Programming by example: Topobo being programmed [7].

Introduction

In the not too distant future computational composites [9] will be everyday design material. By computational composites we mean materials that hold classic material qualities (e.g. structural durability, flexibility, transparency, weight, color, acoustics) but additionally they are capable of sensing (detect changes in their surroundings), actuating (conditionally assume more than one state), and computation. Development within smart materials [1], graphene transistors [8], nanotubes [2] etc. makes these kinds of materials theoretically possible already today albeit not yet practically available. The question is: what would a formgiving practice with those materials look like? How would designers become familiar with the dynamics and different combinations of cause and effect of these materials? Which tools would we need and how would they work? Will we still 'program' them through detached laptops or will the programming happen closer to the materials at hand? We here propose a new research program for developing a material programming practice. Based on an analysis of other physical programming practices combined with our own practices of craft and industrial design we propose a set of qualities we would wish to be supported by a future material programming practice (see Fig 1).

Physical programming

The expressive power of a programming language is constrained by its underlying execution model. A computer program's expressive power – its level of abstraction, potential complexity, and the kind of problems it can solve – is formed by the rules of the language and the execution model it builds on. Textual programming was developed in the context of textual systems. Graphical programming was developed with the graphical interface. And with the rise of physical interfaces, physical programming practices such as tangible programming (TP) and programming by example (PbE) were developed. With the miniaturization of computation and its tight integration into actual material, we need another design and physical programming practice that plays into the same modalities as the materials we design with. Before imagining what such practice could entail, we examine the qualities of two existing physical programming practices TP and PbE.

TP environments [cf. 3; 6] (e.g. Fig 3) use physical objects to represent various programming elements, commands, and flow control structures. The manipulation and arrangement of these objects are then used to construct algorithms [6]. By relying on physical manipulation, TP draws on our bodily experiences. As such it supports spontaneous explorations as well as affords collaboration in context [cf. 4]. The drawback is the particularity it demands – the programming environment is highly task-specific and affords often little in terms of thinking out of the box. Consequently, both the potential action space and the threshold for comprehending the action space are lowered making it suitable for confined application areas such as toys.

PbE [cf. 7] (e.g. Fig 4) is a programming practice where the programmer demonstrates an algorithm to a system by recording a set of actions through an artifact/interface. The actions are then played back in the artifact/interface. PbE is typically applied in situations where the artifact is a one-off, accessible, and tangible, such as in the design of shape-changing interfaces and robots. Like TP, PbE has a low threshold for beginners and non-technical disciplines. Further, its

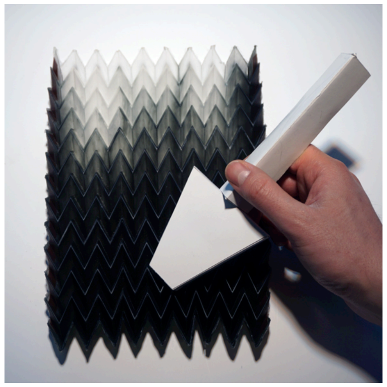
Figure 5 The speculative Select tool used for programming a shape-changing material.

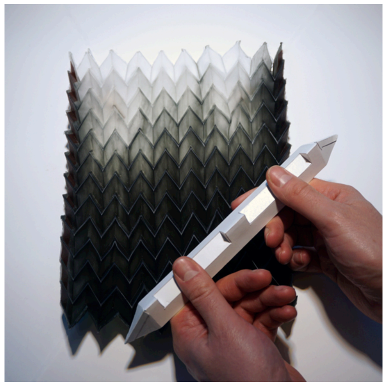
Figure 6 The speculative Force tool used for programming a shape-changing material.

complete lack of abstractions makes composing the behavior immediate. There is no need for the designer to mentally translate conditional and temporal constructs of a code into a behavioral expression. However, the type of behavioral expressions that are possible to compose through programming by example is constrained by what the materials, actuators, and sensors in the artifact allows.

Thus, both TP and PbE hold crucial physical qualities that enables an embodied practice and direct or easy coupling between programming and execution. Where TP allows for at least one layer of abstraction PbE does not. Yet where TP allows for synchronous programming and execution PbE does not. Both, however, tend to be confined to a particular set of artifacts leaving the design space rather limited. With this new research program we wish to explore the possiblities to develop a physical programming practice. Through the use of physical programming tools that works directly on the computational materials we imagine a design practice which blurs the lines between programming and craft.

Material programming

With a continuous interweaving of complex computational technologies and materials over the coming decade [1; 8; 9] it becomes pertinent to develop a programming practice, which enables the designer to stay within the material realm when designing interactive artifacts. This requires "getting a feel" for the dynamics between sensory mechanisms and actuating mechanisms in the materials – of cause and effect. Gaining this "feel" is, however, only really possible through explorations with the materials at hand. Thus we wish to develop a programming practice that supports this. More specifically, we have identified four qualities that a material programming (MP) practice should encompass:

First, an MP practice should not rely on an abstract representation of the programming actions performed on the material. We envision MP to rely on some sort of specified physical tools (see Fig 1, 2, 5 & 6). The tools would be tailored to the unique interactive and physical properties of particular materials, enabling those properties to play a key role in both concept development and actual creation. MP would thus be in line with traditional crafting practices, where several dedicated tools are used for crafting a material, and that can be mastered through practice and skill gained over time. Not unlike a silversmith's practice. A force tool (see Fig 5), for example, could provide the possibility to explore different rhythms and directions of movement in a shape-changing computational composite, supporting an understanding of its properties at hand. The tools would invite and enable the interaction designer to explore the spectrum of possible relations between action (input) and reaction (output) on the material itself.

Second, the physical interaction with the tools and the material enables the designer to slowly develop tacit bodily skills and knowledge. The tools would thus allow the interaction designers to use their body in ways similar to that of crafting non-computational materials, enabling and utilizing the designer's expressive potential and already developed bodily skills (Fig 2).

Third, the practice should unite the 'programming' and 'execution' mode enabling immediacy and a constant focus on the material at hand. The low threshold for exploring the design space thus allows the designer to rapidly prototype a long range of different expressions and interactions. As such, the tools would invite explorations of a materials' potential temporal form, such as rhythms, reaction times, speed, predictability etc. in real time and in-situ.

Fourth, in its hands-on approach the MP practice resembles the practice of PbE with the key difference that MP contains one level of abstraction through the tools. This simple layer allows the designer to freely couple cause and effects in input and output to the degree the material allows. We imagine the possibility of using more advanced algorithms and databases in a back-end design, which would probably rely on traditional textual programming. In that sense MP can be seen as the front-end from a programming perspective.

Finally, a MP practice would not only aid the interaction designer in engaging in a material practice but would also appeal to more traditional design and craft practitioners. Thus our future computational artifacts and environments could be envisioned and designed by people not brought up in technological educations and practices. Consequently, we could also expect a more varied and complex range of expressions and functions. This is a proposal for a new research program, which will seek to unfold, explore, and populate with various degrees of prototypes of MP platforms and tools combined with studies of design and programming practices. As computational composites become more readily available we expect the development of an MP practice would solidify. Until then, we welcome others to join us in exploring the possibilities of a new material programming practice.

References

1. Michelle Addington and Daniel Schodek. 2005. Smart Materials and Technologies. Architectural Press, Elsevier. Oxford, UK
2. Adrian Bachtold, Peter Hadley, Takeshi Nakanishi, & Cees Dekker. 2001. Logic circuits with carbon nanotube transistors. Science. 294, 5545. 1317-1320.
3. Kunal Chawla, Megan Chiou, Alfredo Sandes, & Paulo Blikstein. 2013. Dr. Wagon: a 'stretchable' toolkit for tangible computer programming. In the Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13), pp. 561-564.
4. Ylva Fernaeus and Jakob Tholander. 2006. Finding design qualities in a tangible programming space. In the Proceedings of the Conference on Human Factors in computing systems (CHI'06), pp. 447-456.
5. Felix Hu, Ariel Zekelman, Michael Horn, & Frances Judd. 2015. Strawbies: explorations in tangible programming. In the Proceedings of the 4th International Conference on Interaction Design and Children (IDC '15), pp. 410-413.
6. Timothy S McNerney. 2004. From turtles to Tangible Programming Bricks: explorations in physical language design. Personal and Ubiquitous Computing. 8, 5. 326-337.
7. Hayes Solos Raffle, Amanda J. Parkes, & Hiroshi Ishii. 2004. Topobo: A Constructive Assembly System with Kinetic Memory. In the Proceedings of the Conference on Human Factors in Computing Systems (CHI'04), pp.
8. Frank Schwierz. 2010. Graphene transistors. Nature nanotechnology. 5, 7. 487-496.
9. Anna Vallgårda and Johan Redström. 2007. Computational Composites. In the Proceedings of the Conference on Human Factors in Computing Systems (CHI'07), pp. 513-522.