The Good, the Bad and the Hacked: Creative Coding on Objects

Renaud Gervais
Potioc Lab
Inria Bordeaux, France
renaud.gervais@inria.fr

Jérémy Laviole
Potioc Lab
Université Bordeaux, France
jeremy.laviole@inria.fr

Asier Marzo
Public University of Navarre
Pamplona, Spain
asier.marzo@unavarra.es

Martin Hachet
Potioc Lab
Inria Bordeaux, France
martin.hachet@inria.fr

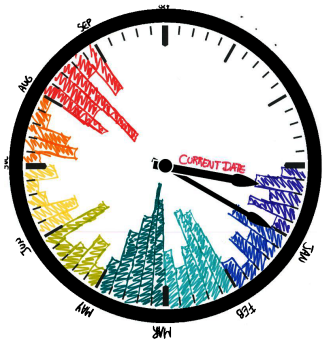

Figure 1: A first draft of a computational clock.

TEI'14, Feb 16 – Feb 19, 2014, Munich, Germany.

Abstract

In a near future scenario, we will replace some of our everyday objects with counterparts in form of Computational Objects (COs). COs look similar to the original object; however, inside them there are input sensors, output devices such as displays and a CPU. Furthermore, COs still convey the context and meaning that the original object had. For instance, a clock is associated with time and thus users could expect its CO version to display time-related data. We suggest that any user should be able to easily code new appearances and behaviors for his or her own objects. Using creative coding as a base, we propose to add the notions of affordances and conventions to this programming context. Moreover, we suggest that COs could be used as a creativity support tool although modifying their behavior beyond conventions could confuse the user. Finally, we reckon that with the proper tools, users could also make physical modifications to COs. For example, a retractile cord can be attached to the clock and be used to pull data out and display them in a linear layout.

Introduction

The Organic User Interfaces (OUI) vision[10, 11] describes a world where everyday objects are equipped with computing capabilities (computational objects – COs). Holman *et al.* [11] defines OUI as: "An Organic User

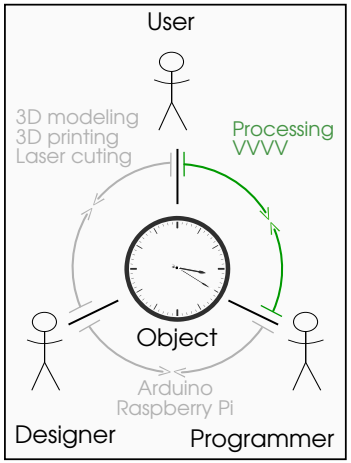

Figure 2: With the maker movement, new emergent technologies enable easy prototyping. In this article we describe the challenge of taking into account the affordances and context to code for these new interactive objects.

Interface is a computer interface that uses a non-planar display as a primary means of output, as well as input. When flexible, OUIs have the ability to become the data on display through deformation, either via manipulation or actuation. Their fluid physics-based graphics are shaped through multi-touch and bi-manual gestures." This vision is made possible by the advances in flexible and curved displays [3] and new touch and sensing technologies [19, 7]. Holman et al. [10] argue that this new reality "raises fundamental questions for the user interface design: how should this organic form be designed?". In this vision, designers have a central role and will need new tools more adapted to organic design. Nonetheless, we think that designers are only one part of the equation. As demonstrated by the rise in popularity of the "maker" culture, users are not only interested in consuming content but also in producing it.

Three main roles can be identified in this situation (Figure 2 *(a)*): the designer, who designs the objects using computational materials; the programmer, that creates the applications running on the COs; and the end-user, who uses the object. The maker culture tends to blur the lines between these roles (Figure 2 *(b)*). We intend to focus on reducing and even bridging the gap between the *programmer* and the *end-user* roles (green section in Figure 2 *(a)*). In this work, we assume the existence of such COs and abstract the physicality (*i.e.* how to build COs) to explore the application development process.

Once computational materials and components have become more widespread, a key aspect to consider will be how users will code for their COs. Each COs has its own sensors and display capabilities. Moreover, unlike a computer or a smartphone, a CO has a use case scenario, context, and history that predates its computing capabilities. Let us consider a computational clock, given its nature the tasks and applications created for it will be time related. Nevertheless, it could also be an internet browser or a calculator, but it could create confusion.

In this article, building on the concepts of smart objects [2] and OUIs [11], we propose to add new uses to COs and we discuss how these new uses will impact the design of such objects. We explore solutions to program COs, their use as a creativity support tool and ways of detecting the boundaries that would deprive COs from their *conventions*.

Related Work

A future populated by computational objects is described by different views and paradigms. The *Internet of Things* is a paradigm in which objects are able to interact with each other and cooperate to achieve common goals [2]. *Radical Atoms* [12] is a vision for human interactions based on a hypothetical physical material that is bidirectionally coupled with a digital model. Consequently, a change in the underlying model is reflected in the physical shape of the object (actuation) and vice-versa. *Organic User Interfaces* (OUI) defined by Holman et al. [11] describe a future where thin and flexible displays and touch sensors will wrap everyday objects. Therefore, any part of an object can be a sensor and/or a display. Holman et al. also presents the concept of hyper-contextualization of the interface to emphasis the fact that some types of COs "should only express a few essential actions, ones that are subject to their form factor" [10]. We position ourselves along the OUI vision although we recognize that these three visions overlap and complement each other and that they may occur in a near (OUI) or distant future (Radical Atoms).

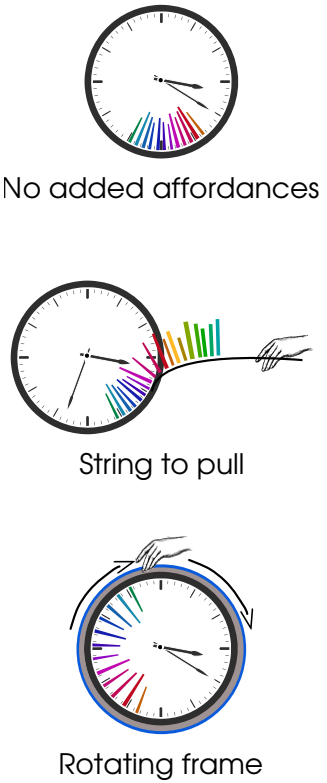

Figure 3: Adding new affordances to a computational clock.

The term *affordance* reflects the actionable properties between the world (object) and an actor (user). Therefore, shape- or color-changing objects also change their perceived affordances. Objects not only convey affordances but also conventions, as introduced by Gibson [6] and Norman [15], conventions are often cultural and learned. Whereas they do not physically prevent an activity, they prohibit some activities and encourage others [16]. SketchSpace [9] is an environment that uses a Kinect and a projector for adding virtual affordances to an object. Thereby, designers can explore the design possibilities of computational materials without the necessity of adding instrumentation such as inertial, touch and proximity sensors. SketchSpace is aimed at assisting object designers during the design iterations.

Reality Editor [8] enables users to edit the behavior of smart objects with a mobile augmented reality system. Users can map inputs and outputs of different devices by directly pointing the Reality Editor to the smart objects and drawing connections between them on the screen. This tool targets end-users whereas we focus more on the gap between users and programmers.

Creative coding is one of the cornerstone of the maker culture. Creative coding toolkits such as Processing [17] or VVVV [1] can help novice programmers and artists. These tools intent to foster creativity by reducing the loop between the code writing and the results. Moreover, they provide high-level interfaces to hardware components such as cameras, Kinects and game controllers. Furthermore, they promote experimentation and reusing of existing code from the community. We consider creative coding as a suitable approach to encourage users to explore and modify the functionality of their COs.

In this work, we build on the pre-existing approach of creative coding and add the notion of *affordances* and *conventions* in the programming framework. Our goal is to help and guide users in their exploration of COs capabilities. We also consider that the user may be the creator of the object, and that he or she could modify its design to create new applications.

Creative Coding on Objects

Everyday objects advance with fashion and their evolution is currently lead by the industrial majors. However, during the next decade, the creation of customized objects with computing capabilities may change our way of consumption. In fact, these highly customizable objects will provide a richer experience than traditional objects. In this section, we explore the existing and new capabilities of a hypothetical computational clock.

Building on existing affordances

The creation of an object is guided by the function of the object and its affordances. Our example of a CO is a clock. Its function will be to display time. It will be equipped with physical hands, a tilt sensor to sense its overall orientation, a circular display and a speaker. It will have preloaded a default application that displays the current time using the physical hands, like any other clock. When a user connects the clock to his development environment, it will expose its affordances (*e.g.* hands can be input or output, screen as output) and conventions (*e.g.* time-related purposes). This will allow the user to quickly explore and experiment with them. Moreover, the clock will expose its current program so that the user is not forced to create applications from scratch.

At this point, the user might want to create a new application that displays his financial transactions of the current year using the clock. He can change the scale of

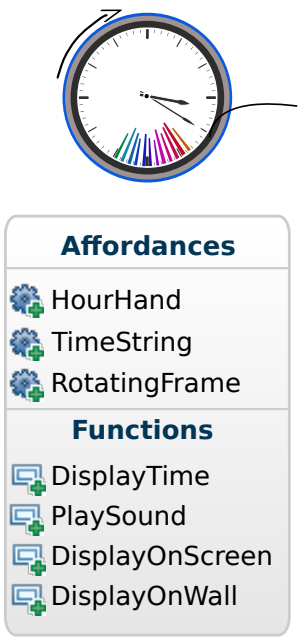

Figure 4: A class diagram exposes the affordances and functions of a CO.

the clock to display the twelve months (January to December) instead of the traditional 1 to 12 numbers. Then, he can create a bar chart based on his financial data, map this chart to a circle and render it on the circular display of the clock. Finally, he can map the position of the hands to change the date of the displayed financial information (as was done by Ishii et al. in the ambientROOM [13]).

Creating new affordances
The direct manipulation of the hands of a clock is not a natural thing to do according to conventions. This issue could appear while modifying other COs. Consequently, we propose the possibility of adding new affordances. The first additional affordance that we propose is to create a rotating frame. It can be used to select a time without touching the hands (Figure 3 *(bottom)*). Another desirable feature would be to display parts of the chart in a linear layout to better compare the values. We could attach a retractable wire on the side of the clock's canvas. When pulled, the wire drags the chart out of the screen and displays it on the wall (Figure 3 *(center)*). When the wire is released, the chart returns to its original shape inside the clock.

Discussion

Exposing Affordances
One approach to expose the functionality of COs is to consider them as programming objects. Therefore, a corresponding class diagram included in the CO instructions will expose its affordances in form of attributes and functions, as illustrated in Figure 4. Nonetheless, this method is close to the programmer view.

Conventions will aid the user to intuit the affordances of the objects that will be available during the programming task; however this only suppose a help. We suggest that visual clues on the CO will create a natural understanding of the possible affordances. For instance, the different output elements can be highlighted with different colors or change their values when the user selects their corresponding attribute. The hand of the clock would glow green and start to rotate quickly when the user chooses this affordance in the help of the object. Correspondingly, for input affordances, the value of its current status can be drawn on them. Incidentally, semantic definition of computational objects may provide richer descriptions of their affordances to the users.

Programming
Similarly to Processing, it is possible to use a 4th generation programming language as the instructions that dictate the behavior of computational objects. Programming languages are proximate to the computer scientist; however, visual coding (*e.g.* vvvv [1]) could remove the necessity of writing code and the possibility of committing syntax errors. It has been shown that children can learn to program using visual blocks that represent code instructions [18]. Moreover, following the idea of Picode [14] images can be used as representations of different states of the computational object. Consequently, users could indicate to the CO the desired behavior in an easier way. Reality Editor [8] propose to program CO by wiring components. In spite of not being an expressive or versatile method, users understand the metaphor easily. It seems interesting to explore other metaphors that facilitate end-user programming.

Programming Environment
The device used to program can mold the way of coding and the user experience. We consider important to explore programming on different settings. Coding on a

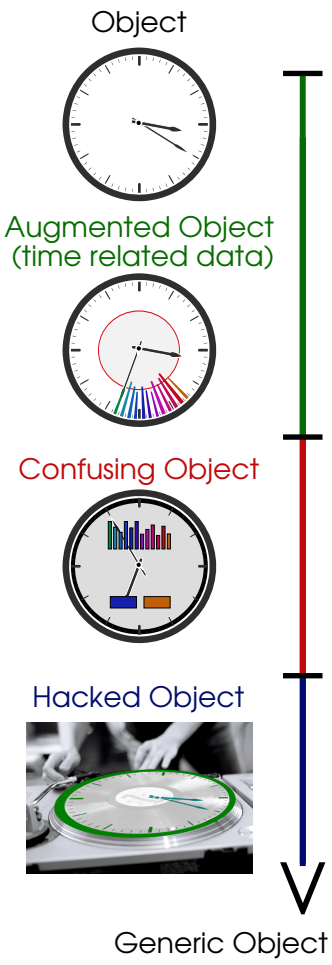

Figure 5: A CO can be either very close to its original purpose (top), have data and functionalities related (middle top) or unrelated (middle bottom) to its original function or be completely hacked away from its original purpose.

Personal Computer seems like the most versatile environment, suited even for power programmers. A Tablet is more portable and appropriate for visual programming. Nevertheless, programming on the surface of the CO or in a projection close to it will situate programming into context.

The Value of COs as a Creativity Support Environment
A computational object could be seen as an expressive and adaptable proving ground for creative exploration. There are specific guidelines designed to measure the creativity support index (CSI) of a tool [5]. In a future experiment we plan to compare the CSI of a computational object and a pure virtual version of it. We expect that affordances will guide the programming experience of the computational object. Thereby, the user will use conventions as creative seeds that mitigate the blank page problem. If a computational object can be the clay in which artists sculpt, a room with several of these objects offers more possibilities and can serve as a full installation. Although conventions could be taken into account by the user, creativity and artistic exploration also reside in the boundaries. Therefore, creative coding of COs may help to explore the border between conventions and discords.

To What Extent Can Functionalities Be Extended Without Losing the Advantages Of Conventions?
Transferring some of the tasks from multipurpose devices such as computers and smartphones to computational objects aims at leveraging users conventions. However, the functionality of the computational object can get out of the scope of its conventions (Figure 5). We reckon that a better understanding of user's conventions is necessary. For doing so, we can measure the usability of a CO as a system [4] while running different applications. Furthermore, it is possible to determine the accuracy of a user describing what an application does, using for instance the applications derived from the creativity support experiment.

Conclusion

Computational objects have the potential to drastically change how we interact with our everyday items. We think that it is worthwhile to develop and implement a view where users are empowered by the ability of shaping the behavior and appearance of the entities that surround them. To this end, we will produce prototypes, using tools from the maker community, to add new affordances to common objects. Then, we will explore how to clearly expose their affordances to the coder and evaluate how COs can stimulate creativity. Finally, we will evaluate the boundaries where an object's design begin to break apart. We think that, given the proper tools, creative users can discover useful and interesting designs of their everyday objects.

References

[1] vvvv. http://vvvv.org.

[2] Atzori, L., Iera, A., and Morabito, G. The internet of things: A survey. *Computer Networks 54*, 15 (2010), 2787 – 2805.

[3] Brockmeyer, E., Poupyrev, I., and Hudson, S. Papillon: Designing curved display surfaces with printed optics. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, ACM (New York, NY, USA, 2013), 457–462.

[4] Brooke, J. Sus-a quick and dirty usability scale. *Usability evaluation in industry 189* (1996), 194.

[5] Carroll, E. A., Latulipe, C., Fung, R., and Terry, M. Creativity factor evaluation: Towards a standardized survey metric for creativity support. In *Proceedings*

of the Seventh ACM Conference on Creativity and Cognition, C&C '09, ACM (New York, NY, USA, 2009), 127–136.

[6] Gibson, J. J. *The ecological approach to visual perception.* Houghton, Mifflin and Company, 1979.

[7] Harrison, C., Benko, H., and Wilson, A. D. Omnitouch: Wearable multitouch interaction everywhere. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, ACM (New York, NY, USA, 2011), 441–450.

[8] Heun, V., Hobin, J., and Maes, P. Reality editor: Programming smarter objects. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, UbiComp '13 Adjunct, ACM (New York, NY, USA, 2013), 307–310.

[9] Holman, D., and Benko, H. Sketchspace: designing interactive behaviors with passive materials. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, ACM (New York, NY, USA, 2011), 1987–1992.

[10] Holman, D., Girouard, A., Benko, H., and Vertegaal, R. The design of organic user interfaces: Shape, sketching and hypercontext. *Interacting with Computers 25*, 2 (2013), 133–142.

[11] Holman, D., and Vertegaal, R. Organic user interfaces: designing computers in any way, shape, or form. *Commun. ACM 51*, 6 (June 2008), 48–55.

[12] Ishii, H., Lakatos, D., Bonanni, L., and Labrune, J.-B. Radical atoms: Beyond tangible bits, toward transformable materials. *interactions 19*, 1 (Jan. 2012), 38–51.

[13] Ishii, H., Wisneski, C., Brave, S., Dahley, A., Gorbet, M., Ullmer, B., and Yarin, P. ambientroom: Integrating ambient media with architectural space. In *CHI 98 Cconference Summary on Human Factors in Computing Systems*, CHI '98, ACM (New York, NY, USA, 1998), 173–174.

[14] Kato, J., Sakamoto, D., and Igarashi, T. Picode: Inline photos representing posture data in source code. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, ACM (New York, NY, USA, 2013), 3097–3100.

[15] Norman, D. A. *The psychology of everyday things*. Basic books, 1988.

[16] Norman, D. A. Affordance, conventions, and design. *interactions 6*, 3 (May 1999), 38–43.

[17] Reas, C., and Fry, B. *Processing: a programming handbook for visual designers and artists*, vol. 6812. Mit Press, 2007.

[18] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., and Kafai, Y. Scratch: Programming for all. *Commun. ACM 52*, 11 (Nov. 2009), 60–67.

[19] Sato, M., Poupyrev, I., and Harrison, C. Touché: Enhancing touch interaction on humans, screens, liquids, and everyday objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, ACM (New York, NY, USA, 2012), 483–492.