Mapping Objects with the Surface Editor

Alain Crevoisier
Haute Ecole de Musique de Genève (HEM)
Rue de l'Arquebuse 12, CP 5155
CH-1211 GENEVE 11
alain.crevoisier@hesge.ch

Cécile Picard-Limpens
Haute Ecole de Musique de Genève (HEM)
Rue de l'Arquebuse 12, CP 5155
CH-1211 GENEVE 11
ccl.picard@gmail.com

ABSTRACT

The Surface Editor is a software tool for creating control interfaces and mapping input actions to OSC or MIDI actions very easily and intuitively. Originally conceived to be used with a tactile interface, the Surface Editor has been extended to support the creation of graspable interfaces as well. This paper presents a new framework for the generic mapping of user actions with graspable objects on a surface. We also present a system for detecting touch on thin objects, allowing for extended interactive possibilities. The Surface Editor is not limited to a particular tracking system though, and the generic mapping approach for objects can have a broader use with various input interfaces supporting touch and/or objects.

Keywords

NIME, mapping, interaction, user-defined interfaces, tangibles, graspable interfaces.

1. INTRODUCTION

Projects on tangible interfaces [11] have grown since the last decade, and a large part of them concerns new ways of performing music [4]. More recently, the concept of Natural User Interface (NUI) has been used and refers to "*a user interface that is effectively invisible*" [7]. Our work gathers achievements in the field of graspable interfaces, NUI technologies and new interfaces for musical expressions. We extended and adapted our previous research on tactile interfaces for the use of graspable objects, leading to a concept of interaction based on combining the manipulation of objects with touch sensing. Detecting fingers touching the surface of an object offers the opportunity to intuitively trigger actions by simply tapping on the object. For our experiments, we have used a multitouch technology that makes possible to transform any flat surface into a multitouch device [1] and ReacTIVision, a well known Computer Vision tool for identifying objects and tracking their position and orientation using visual markers [3]. However, the framework we have developed for mapping objects is not relying on a particular technology and any input interface supporting the TUIO protocol [5] can be used, although the touch-on-object information may not be available in all cases.

Complex mapping structures can be determined with the Surface Editor thanks to the possibility to assign several actions for an object and also to set rules for the conditional activation of an action or a group of actions [6]. This is particularly useful for exploring new mapping strategies between input gestures and musical actions. The finality of our study is to propose a generic mapping tool for setting up and configuring graspable interfaces adaptable for any use case scenarios.

2. RELATED WORK

A new trend in the field of Human Computer Interaction (HCI) has been observed this last decade: interfaces are more and more adapted to our ways of experiencing the world. As an example, multitouch interaction is gradually supplanting the use of the traditional computer mouse as control component. Another example is given by graspable interfaces. As noticed in 1995 by Fitzmaurice et al. [2], a graspable interface exploits not only our well-developed, everyday haptic-tactile skills for physical object manipulation, but also our sharp spatial reasoning capacities. In addition, it enables multi-person, collaborative use. Playing with objects, combining them in order to create something new, is a way of showing our interpretation of the world [13]. Further, the use of objects as controllers give a persistent representation of what is manipulated [8]. The idea is not only to handle objects independently from each other but also putting them into relation with one another. As outlined by Wanderley et al. [12], performing computer music with controllers is closely related to the notion of mapping. Indeed, analyzing the influence of mapping on the performance of digital musical instruments or systematically defining mappings to relate controller variables to synthesis inputs remain crucial.

Among the many tangible interfaces projects that have been developed over theses two decades [4], the Reactable [3] is by far the most remarkable, and the closest to our study. The Reactable takes its origin from studies on musical performance and the interest in developing interfaces for the real-time creation and exploration of music. It uses visual marker recognition (VMR) for object identification. The specific set-up of the ReacTable includes a semi-transparent surface with a camera and projector behind it.

Our method differs in many ways. First of all, it uses ordinary tables and surfaces for interaction. Custom made tables can be visually attractive, but they have the main drawback of not being suitable for widely use. Second, contrary to the majority of similar projects that require an image to be projected on the table, it is only an option in our case. This makes the system easily transportable everywhere. Already with one of the first projects on tangible interfaces, Audiopad [9], cumbersome video projection was seen as a technical limitation. Third, none of similar projects allows for detecting fingers touching on an object. In that way, our system gives the opportunity to explore new interactive possibilities. Finally, as pointed by the SenseTable project [8], interacting with a large amount of information with a finite number of physical objects remains challenging. For this purpose, we propose to leave to the user the ability to define and adjust the actions in response to his interaction, so that he can design the mapping that fits his need.

3. OBJECT DEFINITION

An object is defined as any element that can be grasped and put on a surface. Objects must be identified somehow. Most systems require a tag to be attached to the objects in order to facilitate the object recognition and identification. Objects are defined by their attributes: Object ID (Tag ID), Size, Tag position, and Type. The last attribute is useful to set families of objects. It is up to the users to give a meaning and function to the objects. For instance, one could attribute characteristics related to sound and music to objects, such as sound sources, sound modifiers (effects), loop players, volume button, track selector, or any kind of processing parameter.

4. OBJECT RELATIONS AND DERIVED PARAMETERS

4.1 No Relation

In this case, the object do not relate with any other object, even if there might be some others in the neighborhood. The only parameters are the absolute position and rotation angle of the object. These may be considered as the intrinsic parameters of an object, since they exist in all cases, independently of its relation with other objects.
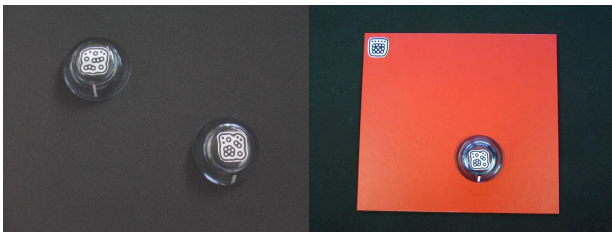

Figure 1. Neighborhood relation (left) and Object-in-Object relation (right).

4.2 Neighborhood

This is the simplest relation between two objects (Figure 1, left). Beside the intrinsic parameters peculiar to each object, new parameters are derived from the relative position of the two objects:

- Delta X   - Distance
- Delta Y   - Angle

Except for the angle between the two objects, which is always calculated from center-to-center, the other parameters are calculated both from center-to-center and from edge-to-edge, providing a total of seven derived parameter.

4.3 Object in Object

This relation exists for instance if a thin object is sufficiently big in size to contain one or more smaller objects (Figure 1, right). In this case the derived parameters are given by the relative position of the smaller object inside the bigger one.

4.4 Selective Relations

In most cases, we don't want an object to relate with all others. For instance, we may want that an object can relate only with a particular kind of objects, or only within a certain distance. In order to consider selective relations, we need to define conditions under which two or more objects can enter in relation. For this purpose, we introduce the notion of *filters*. Filters can be applied either to the Object ID, the Object Type, the intrinsic parameters, or the derived parameters. In addition, several filters can be combined to define more selective conditions.

5. CHAINS

Chains are created when two or more objects enter in relation. They can also be seen as sequences of objects ordered in the 2D space. Figure 2 shows two examples of chains where objects are close to each other. However, objects do not need to be close to form a chain and the two configurations in Figure 1 are also valid examples of chains. A chain exists as long as the objects are in relation and that the conditions set for the filters are satisfied. A chain is defining a new entity, which extends the object's characteristics. As a consequence, objects belonging to a chain get new attributes:

- The ID of the chain they belong to (Sequence ID)
- Their position in the chain (Ordering number)
- The number of elements in the chain (Chain length)

For simplicity, we did not individuated branches in a chain. For this reason, several objects may get the same ordering number (see section 5.2 for details on attribution).
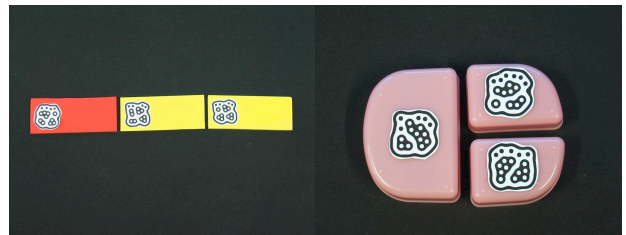

Figure 2. Two examples of chains.

5.1 Master/Slave Objects

Let's imagine a situation where an object would represent a track and a second object a clip to play in this track. The two objects do not have the same hierarchical level since it is the second object that must inherit the Sequence ID from the first one. For this reason, objects have an additional attribute in order to determine if they are Master objects or Slave objects. Master objects will hold a Sequence ID, and slave objects will inherit this Sequence ID when they enter in relation with the master object. If a chain is formed only with slave objects, then they will not get any Sequence ID. Similarly, they will not get an Ordering number since their position in the chain is calculated respectively to the master object (Figure 3, top). Master objects cannot enter in relation, even if their selective filters would allow it. But it can happen that a slave object would relate with more than one master object (Figure 3, bottom). In this case, there are rules of exclusion based either on the order of occurrence (first master-slave relation will supersede any further relation), or on a spatial distribution (for instance, only the leftmost relation will be considered).
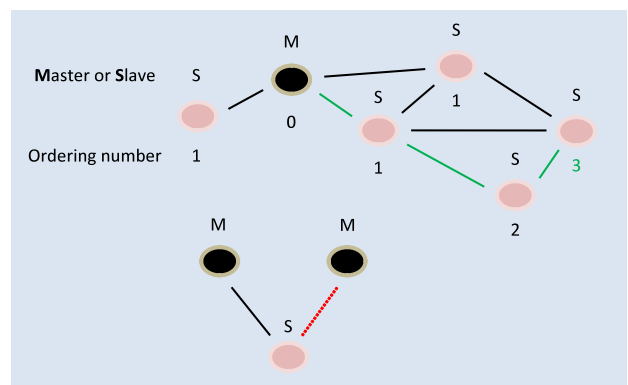

Figure 3. Ordering number attribution (top) and Multiple Slave-Master relation exclusion (bottom).

5.2 Attribution of the Ordering Number

When several objects are in relation in a chain, it is frequent that closed loops are created. In practice, it is better to avoid confusing relationships by setting the appropriate filters between objects. However, we had to find a way to manage any possible situation. For each slave object, the ordering number algorithm searches for a path towards the master object with the shortest distance between neighbors (Figure 3, top). The order is then the number of steps necessary to reach the master object. An additional rule imposes that a direct relation with the master object will supersede any other relation.

Table 1 is giving a summary of all the attributes for an object and if they are defined at the design level (by the user) or at the performance level (by the system).

Table 1. Summary of objects attributes

Attribute	Design or Performance Level
Object ID (= tag ID)	Design
Object size	Design
Type	Design
Tag position	Design
Master or Slave	Design
Sequence ID	Design or Performance
Ordering number	Performance
Chain length	Performance

6. EVENTS

In order to trigger actions when objects are manipulated by users, for instance to trigger loops or to vary sound processing parameters, it is necessary to generate triggering events. They are linked to the input parameters provided by the tracking system (intrinsic parameters and derived parameters), and occur when certain conditions are met. In this study, we consider two families of events, Object Events and Touch Events. We also make a distinction between *discrete* events and *continuous* events present in both families.

6.1 Connect Event

A Connect event is a discrete event that occurs when an object enters in relation with another one. The second object may be either alone or within an already existing chain of objects. The conditions for a Connect event to occur are specified by the filters defined in section 4.4. For instance, an object could generate a Connect event only when it is close enough to the right side of another object of the same type. This would require three filters, one setting the type, one setting the distance, and one setting the range of the valid angle between the two objects.

6.2 Other Object Events

In addition to the Connect event, other object events include:
- Object Down: the object is placed on the surface.
- Object Up: the object is removed from the surface.
- Object Exists: the object is on the surface (continuous).
- Object Moving: position or angle is changing (cont.).
- Object Moved: position or angle has changed (discrete).

6.3 Touch Events

If the tracking system is capable of detecting when objects are being touched, then additional events are available for triggering actions:

- Touch Down: the object is touched.
- Touch Up: the object is stopped being touched.
- Drag Start: a dragging movement starts on the object.
- Drag Stop: the dragging movement stops.
- Touching: lasts while the object is touched (cont.).

Filters are also available in this case for additional selective conditions [6].

7. IMPLEMENTATION

The Surface Editor has been considerably extended in order to support objects management and also to integrate more closely with Ableton Live. A new class of mapping components have been added (Objects), a new activator has been created to handle objects (Object Activator), and two new actions have been added specifically for Ableton (Live Track and Live Device). Also, it is now possible to send variables between objects and controllers in order to change the behavior of an object or controller from another one. This can be used, for instance, to change the MIDI channel of an object's action from the rotation angle of another object. Finally, controllers and objects can be used together for additional flexibility.

Figure 4. Setup.

The Surface Editor receives the touch and object information via TUIO provided by two tracking systems running in parallel. The first one is the Airplane controller developed in previous projects [1], and the other one is a simple webcam hooked to ReacTIVision [10]. Thus, two cameras are looking to the scene (Figure 4, right). A video projector is also used for the optional projection of visual feedback on the table.

Since the Airplane controller is detecting touch by watching fingers crossing a plane of IR light placed a few millimeters above the surface, objects must be thin enough for not interfering with the plane (Figure 5). If interacting with touch gestures is not desired, then thicker objects can be used.

7.1 Linking Controllers to Objects

Controllers rely on a graphical representation, like a fader or keyboard for instance, and need a visual display in order to be manipulated. On the other hand, as mentioned before, objects hold a persistent representation and do not need a display for visual feedback. However, if projecting an image is not an issue, it can be desired to combine the two interaction paradigms. For this reason, several options exist with the Surface Editor. First, it is possible to arrange controllers on a page and leave some blank space in order to use objects at the same time. However, using pages is a rather static approach compared to the manipulation of objects. In order to bring a more dynamic dimension to the use of controllers, it is now possible to link a controller, or group of controllers to an object. Once linked, the controller(s) will appear dynamically when the

corresponding object is placed on the surface (see Figure 5). Moving the object will also move the controller accordingly.

Figure 5. Linking Controllers to Objects.

7.2 Object Activator

The new Object Activator implements the triggering mechanism described in section 6 and the selective filters described in section 4.4. The activator is therefore determining the conditions for an action, or group of actions to occur. Users first select the triggering event and then add as many filters as desired (Figure 6). Like with the Touch Activator available so far with the Surface Editor, several Object Activators can be set for an object, so that several actions or group or actions can be triggered according to different conditions.
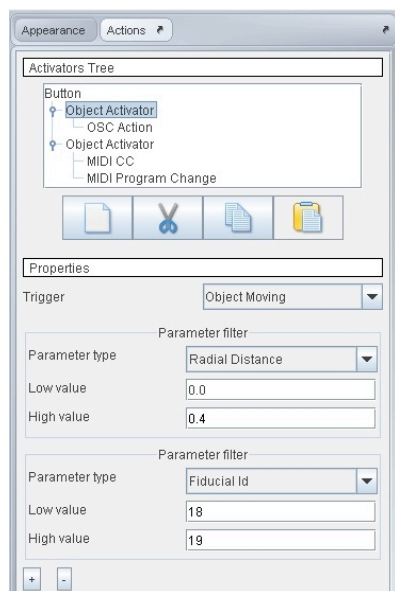

Figure 6. Control panel of an Object Activator.

7.3 Sound Generation

The Surface Editor is now supporting LiveOSC[1] in order to perform bi-directional communication with Ableton Live. This is simplifying enormously the mapping work compared to MIDI. The Surface Editor is informed of the clips, volume, and devices present in a track with all their parameters. Instead of a double work, setting a MIDI CC on one side and setting the same on the other side for the parameter one desires to control, users can map a parameter in Live simply by selecting it in a dropdown list. Two new actions have been created, one to handle track related features (clip, volume, mute, etc.), and one to handle device related features (device parameters).

The Surface Editor is of course not limited to interact with Ableton Live. All variables and information events can be sent through OSC or MIDI to any other sound generation software.

---

[1] http://liine.net/livecontrol/ableton-liveapi/liveosc/

8. FUTURE WORK

Several user evaluations are planned. First, with novice users using pre-defined scenarios, in order to test the suitability of this system for music pedagogy. Second, the platform will be introduced to more advanced users during a three days workshop that will be held between April 22 and 24 2011, in the context of the Electron Festival[2] in Geneva, Switzerland.

9. ACKNOWLEDGMENTS

10. REFERENCES

[1] Crevoisier, A., and Kellum, G. Transforming Ordinary Surfaces into Multi-touch Controllers. *Proc. of International Conference on New Interfaces for Musical Expression (NIME)*, 2008.

[2] Fitzmaurice, G. W., Ishii, H., and Buxton, W. Bricks: Laying the Foundations for Graspable User Interfaces. *Proc. of Conference on Human in Computing Systems (CHI'95)*, 1995.

[3] Jordà, S. and Kaltenbrunner, M. and Geiger, G. and Bencina, R. The reacTable*. *Proc. of ICMC*, 2005.

[4] Kaltenbrunner, M. http://ww.iua.upf.emtg/*reactable*/ ?related , Referenced October 20, 2006.

[5] Kaltenbrunner, M., Bovermann, T., Bencina, R. and Costanza, E. TUIO - A Protocol for Table Based Tangible User Interfaces. *Proc. of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation* (GW 2005), Vannes (France).

[6] Kellum, G., and Crevoisier, A. A Flexible Mapping Editor for Multi-touch Musical Instruments. *Proc. of NIME-09*, Pittsburgh, USA, 2009

[7] Natural User Interface: http://en.wikipedia.org/wiki/Natural_user_interface

[8] Patten, J., Ishii, H., Hines, J., Pangaro, G. Sensetable: A Wireless Object Tracking Platform for Tangible User Interfaces. Proc. of *CHI'01*, ACM Press, pp.253-260, 2001.

[9] Patten, J., Reht, B., and Ishii, H. Audiopad: A Tag-based Interface for Musical Performance. *Proc of NIME-02*, (2002), 24-26.

[10] ReactiVision: http://reactivision.sourceforge.net/

[11] Ullmer, B., and Ishii, H. Emerging frameworks for tangible user interfaces. IBM Systems Journal 39 (2000), pp. 915–931.

[12] Wanderley, M., and Depalle, P. Gestural Control of Sound Synthesis. *Proc. of the IEEE*, vol. 92, No. 4 (April), Special Issue on Engineering and Music - Supervisory Control and Auditory Communication, G. Johannsen, Ed., pp. 632-644.

[13] Wolf, M. Soundgarten: A Tangible Interface that Enables Children to Record, Modify and Arrange Sound Samples in a Playful Way. *Masters thesis*, University of Applied Sciences Cologne, Germany, 2002.

11. ADDITIONAL RESOURCES

More info and videos at: www.future-intruments.net

---

[2] http://www.electronfestival.ch/