# Dynamic Patches for Live Musical Performance

Martin Kaltenbrunner
Music Technology Group, IUA
Universitat Pompeu Fabra
Ocata 1, 08003 Barcelona, Spain
+34 93 542 2104
mkalten@iua.upf.es

Günter Geiger
Music Technology Group, IUA
Universitat Pompeu Fabra
Ocata 1, 08003 Barcelona, Spain
+34 93 542 2104
ggeiger@iua.upf.es

Sergi Jordà
Music Technology Group, IUA
Universitat Pompeu Fabra
Ocata 1, 08003 Barcelona, Spain
+34 93 542 2104
sjorda@iua.upf.es

## ABSTRACT
This article reflects the current state of the reacTable* project, an electronic music instrument with a tangible table-based interface, which is currently under development at the Audiovisual Institute at the Universitat Pompeu Fabra. In this paper we are focussing on the issue of Dynamic Patching, which is a particular and unique aspect of the sound synthesis and control paradigms of the reacTable*. Unlike common visual programming languages for sound synthesis, which conceptually separate the patch building process from the actual musical performance, the reacTable* combines the construction and playing of the instrument in a unique way. The tangible interface allows direct manipulation control over any of the used building blocks, which physically represent the whole synthesizer function.

## Keywords
Tangible Interfaces, Musical Instrument, Sound Synthesis, Visual Programming, Dynamic Patching

## 1. INTRODUCTION
The reacTable* is thought to be an electro-acoustic musical instrument in the tradition of Jorda's FMOL synthesizer [1]. The main idea is the creation of a tangible electronic musical instrument that allows expressive collaborative live performances for professional musicians without the limits of many screen-based interfaces for electronic music. As its name suggests the reacTable* is a table-based instrument; it can be played by manipulating a set of objects that are distributed on top of the table surface. Each of these objects has its dedicated function for the generation, modification or control of sound. Bringing these objects into proximity with each other constructs and plays the instrument at the same time. While the table is equipped with sensors for the identification and tracking of the objects, the players themselves do not (and must not) have to wear any controller devices. In addition to the sound which is obviously produced while playing, the reacTable* also provides additional visual feedback by projecting a visualisation of the sound-flow onto the table surface. This creates a truly multi-modal musical experience incorporating all possible senses. Due to the large dimension of the instrument – a round table with a diameter of around 1.5m – the reacTable* is intentionally designed to be playable by more than one person at a time. These collaborative aspects will be taken even further by networking two or more instruments allowing remote collaboration. For a more detailed description of the various aspects of this instrument see Sergi Jordà's last year's papers which introduced the general reacTable* concept [2][3].

## 2. CURRENT STATE
During the first project phase we have been developing the basic reacTable* concepts within a software prototype only, simulating the tangible user interface component with a graphical interface. This approach allowed the rapid prototyping and the introduction of new synthesizer and interaction elements without worrying about sensor and hardware problems beforehand. In the second phase the GUI simulation was replaced by an actual computer vision module, which allowed the identification and tracking of objects using simple visual markers. This has led to the construction of a first tangible prototype, which also takes into account the design of the physical objects; this topic related to the object-to-sound mappings is discussed elsewhere in a more detailed manner [4].

The current system was implemented in a completely modular way, allowing the easy reuse or replacement of the four basic functional components: a sensor module is tracking the state, position and orientation of any object that is currently present on the table. These raw sensor parameters are passed to the central management component, which drives the two actual synthesis components for the sonic and graphical feedback. All these components are completely independent and are communicating via a simple proprietary network protocol (which we consider to upgrade to OpenSound Control [5] compatibility if necessary). This separation allows the execution on various different hardware platforms avoiding eventual performance bottlenecks since each of these modules actually requires quite a lot of computational resources.

In this paper we are mostly concentrating on the various functions of the central management component, but first we are going to provide a quick overview on the current implementation state of the further components:

### 2.1 Computer Vision
The current tangible prototype is incorporating Costanza's D-touch vision engine [6], using simple visual symbols, which can be easily printed and attached to the objects. This system allows the tracking of the object's position and orientation at a reasonable frame-rate (7fps at 640x480 using a 1GHz System). The recognition performance of this vision engine is relatively robust due to the concurrent development of the marker symbols and the recognition algorithm. The apparent drawback of this approach is the visibility of the markers, which partly has been overcome by attaching them onto the objects' bottom side. We are also using back-vision in order to avoid hand and body occlusion problems.

## 2.2 Sound Synthesis

The sound synthesis is currently implemented using Puckette's Pure Data visual sound programming language [7][8]. Through its internal message system Pd can be fully controlled from the outside; most operations that are accessible through the user interface can be addressed by these messages. We had to extend Pd to allow the disconnection of objects through these internal messages. These changes meanwhile found their way into the official Pd version.

The sound engine implements a large set of basic synthesis and control components, which can be instantiated at startup in any desirable number. During runtime this engine is receiving simple control messages from the management unit redefining the sound and control connections within this network of processing objects.

The processing objects are higher level objects implemented as abstractions, they are built using smaller low level units that are available in Pd. Examples of audio objects are various analog style oscillators, wave-table oscillators and granular synthesis. Each of the tangible reacTable* objects has its direct counterpart in an abstraction implemented with Pd.

## 2.3 Vision Synthesis

As Jordà pointed out in earlier presentations [2][3], visual feedback is a fundamental component of the reacTable*. First informal tests indeed show that this visual feedback is actually crucial for the playability of the instrument.

Central feature of this visual feedback is the visualization of the sound-flow between the processing objects, basically a representation of the waveform state at each object. In a similar way we visualize the control messages and the object state. The players can control the instrument by either manipulating the objects themselves or the sound flow representation in between. A simple gesture to illustrate this principle is the cut or muting of a sound stream, which is done with a karate-style hand gesture. We have been working on various concept studies on the visualization of sound and control flow and object activation auras. These will be made available as various *skins* or visual themes, which can be chosen for an individual session.

Similar to the camera setup, the visual feedback is projected from below to avoid disturbing hand and body occlusions.

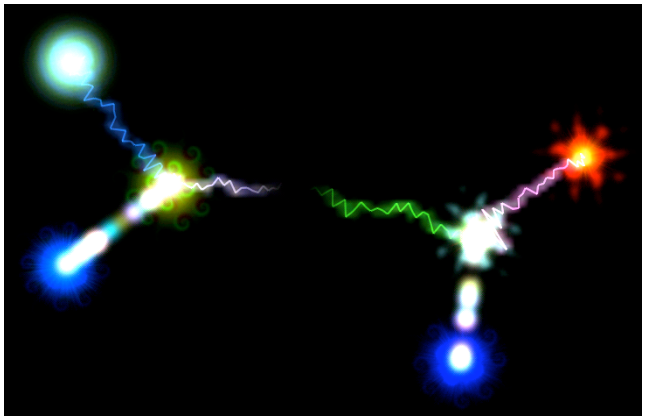

**Figure 1: Dynamic Patch Visualization.**

# 3. THE TANGIBLE INTERFACE

The central feature of the reacTable* is its object based tangible interface, in which virtual synthesis building blocks are represented by corresponding physical objects, which can be directly manipulated by the performers.

## 3.1 Object Handling

A set of objects, which are made available on the table, can be manipulated by the players in various ways: Once put onto the table the objects are activated, the objects can be moved on the table surface - bringing the objects in relation to each other. The rotation angle of the object is tracked as well.

Most reacTable* objects are plain and passive, meaning that they do not come with any cables, switches buttons or whatsoever. The user also does not have to wear any special sensor or controller equipment for the object handling; the plain hands are the only necessary controller. This of course, does not rule out the possibility of smart objects that incorporate additional internal electronics in order to retrieve some additional sensor data, coming from squeezing, bending or bouncing them, like in the case of Weinberg's Squeezables [9]. In any case, this has to be achieved in a completely transparent way, using wireless technology for example, so that the performer can treat all objects in an equal way. A rubber hose or a wooden toy snake, whose state could be either determined by the computer vision or by using some bending sensors like in the Sonic Banana [10], can serve as an advanced controller producing multi-dimensional control data.

## 3.2 Gesture Control

The hands play an important role: They not only can manipulate the reacTable* objects, they are treated as a super-object themselves. We also track the position and state of the hands in order to retrieve additional control data. A nice example is the wave-table object, which simply can be programmed by finger-painting a waveform close to it. This painted waveform is "absorbed" by the object that starts playing immediately. As already mentioned above, the hand can also control the visual sound representation by cutting or redirecting the sound flow.

# 4. DYNAMIC PATCHES

## 4.1 Object Types

The reacTable* objects can be generally categorized into seven different functional groups: Generators, Audio Filters, Controllers, Control Filters, Mixers, Clock synchronizers and Containers. There are also some exceptions that do not fit within any of these categories.

- *Generators* are sound sources that can produce various types of synthesized or sample based sound. They have an audio output and various control inputs.
- *Audio Filters* can modify incoming sound based on their internal algorithms, which can range from a simple low-pass filter to any possible sound effect. Filters have generally one or two sound inputs and a sound output as well as several inputs for control.

Control inputs permit the constant modification of the object parameters. In the reacTable*, parameters for all kinds of

objects can be modified either by changing the physical object spatial properties (e.g. position, orientation, distance to the next object, angle to the next object, distance to the center, angle to the center, etc.), in some cases even its morphological properties (e.g. bending, shape…), or by connecting control data flows to their control inputs. These data flows are generated by a third type of objects, the Controllers.

- *Controllers* generally produce their control data by algorithmic generation, which include simple low frequency oscillators up to complex chaotic or fractal generators. Like in any other object, their respective parameters (e.g. frequency and range in a low frequency oscillator) also depend on the spatial properties of the object, and can be permanently modified. Controllers do not yet have inputs but we plan to implement this feature soon. With some exceptions, controller output is generally adimensional, which means that the effect of a controller depends on the control input it connects to.
- *Control filters* process control data. They have a control input and a control output, and unlike regular controllers, their output can sometimes be dimensional; the output values of a harmonizer or a chord generator, for example, are always mapped to pitch.
- The *Mixer* object can take various sound streams as an input and produces a single output stream. Inverted Mixers (*Splitters*) split a single sound into multiple output streams.
- *Clock synchronisers* introduce a higher hierarchy; they can influence several objects on their proximity at once and in several ways, like sending synchronised triggers or correcting low frequencies in order to match a given pulsation. Clock synchronisers have one fundamental parameter, tempo (they also have *tempo subdivision*), which can be modified by repeatedly hitting the object.
- Finally there are the higher-level container objects, which virtually can contain a pre-built set of sub-patches, allowing the construction of more complex sound structures.

## 4.2 Connection Rules

Dynamic patching does not require the user to explicitly connect the objects. We defined a simple set of rules, which automatically connect and disconnect objects. As mentioned above all objects have a certain number of two different in-output connectors: Sound and Control. Based on a simple distance rule, each object checks its neighborhood for objects, which can provide both compatible and available ports. It will therefore always choose the closest available object. We are currently working on some additional connection rules, such as the introduction of pseudo-physical forces like “magnetic fields”.

The reacTable* connection paradigm produces a highly dynamic environment. Moving an object around the table surface permanently interferes and alters existing connections, creating extremely variable synthesizer morphologies. In order to avoid this not always desirable destructive behavior we have introduced two features. Lifting an object from the table surface immediately deactivates it and it ceases to interfere with nearby objects. Additionally, we introduced an explicit linking gesture, which permanently links two objects. Bringing two objects together so that they touch each other, establishes a permanent link between both, which is only broken when one of the two objects is lifted.

## 4.3 Building vs. Playing

Within traditional modular visual programming synthesizers, there is a clear separation between building and playing the patch (or instrument): There is an editing and an execution mode. The editing is usually a lengthy development process, which leads to a final and stable instrument patch, which then during the execution mode is controlled on screen or via any available controller device. The reacTable* has to be built and played at the same time. Each piece has to be constructed from scratch starting from an empty table (or from a single snapshot which has been (re)constructed on the table before the actual performance). This is a fundamental characteristic of this instrument, which therefore always has to evolve and change its setup. Building the instrument is equivalent to playing it and vice-versa, and remembering and repeating the construction of a building process can be compared to the reproduction of a musical score. This also poses some conceptual constraints, which are difficult to surpass.

## 4.4 The Container problem

As shown above, one could imagine some pre-built and saved setups, but due to the physical nature of the instrument this leads to certain problems. If we want to recall a previously saved snapshot of the object arrangement on the table we have basically two options: Project the previous object types and positions onto to the table surface and wait until the performer has re-arranged the objects in the correct way, and then continue from the recorded point. We could also imagine that the saved state continues to be only represented by non-physical projected object representations, which of course cannot be directly manipulated, but which would have at least the same behavior as the physically present objects. The Music Table [11] manages to overcome this problem with the introduction of an additional controller object that is used to manipulate these virtual objects.

A similar problem arises with the introduction of sub-patches, a common technique in visual programming languages. For recovering and reusing pre-constructed setups, we have devised a container object, which can embed any portion of the reacTable*. We are currently working on a gesture (drawing a circle around a certain table region and point to an available container object), which will allow to virtualize and save parts of the current setup. Although this is intuitive and simple enough, we still will have to get rid of the now deactivated physical objects that remain on the table.

# 5. SOME HISTORICAL BACKGROUND

Several tangible interfaces and systems have been designed with the purpose of taking advantage of the richness of multimodal human senses and skills developed through our lifetime of interaction with the physical world [12]. Some of them like the SmallFish [13], the Jam-O-Drum [14][15], the Musical Trinkets [16], Augmented Groove [17] or the Audiopad [18] are indeed musical applications.

However, in the reacTable*, three additional concerns are no less important than tangibility: modular synthesis, visual programming and visual feedback. The concept of modular synthesis goes back to the first synthesizers, both in the digital (Max Mathew’s Music-N) [19] as in the analog domains (with Robert Moog’s or Donald Buchla’s Voltage-controlled

synthesizers) [20]. Modular synthesis has largely proved its unlimited sound potential and has been indeed the essential element of all the visual programming environments for sound and music, which started with MAX in the late 80s [21], and have developed into PD [8], JMax or Reaktor, to mention a few. As shown by all of these healthy environments, visual programming constitutes nowadays the more flexible and widespread paradigm for interactive music making. Visual feedback on its turn, understood as a tool for maximizing the player-instrument bandwidth, is a more recent concept that is just starting to show its potential [2][3][22]. In that sense, the reacTable* is probably the first system that seeks to incorporate all of these paradigms, in order to build a flexible, powerful and intuitive new music instrument.

## 6. FUTURE WORK

The work on the reacTable* is progressing constantly. We already have a working prototype, which still differs though in various aspects from the original concept. Much of the future work will go towards three major directions: We are considering the introduction of a dual sensor system, by introducing RFID tags for the general object tracking. Computer vision will be mainly used for the hand gesture recognition and the tracking of untagged objects. A lot of the ongoing work on the visual feedback is going to be included into the working prototype in the near future and we have been working intensively on the object design and mapping issues, which will also be reflected in the final instrument design.

## 7. ACKNOWLEDGMENTS

We would like to thank the following people for their various contributions to the current reacTable* prototype: Enrico Costanza from Media Lab Europe, Dublin for providing his CV engine. And all the other collaborators of our Interactive Sonic Systems team at UPF: Alvaro Barbosa, Ruben Hinojosa, Ignasi Casanovas, Carlos Manias and Xavier Rubio.

## 8. REFERENCES

[1] Jordà, S.: FMOL: Toward User-Friendly, Sophisticated New Musical Instrument. Computer Music Journal. Vol.26.3 pp 23-39, 2002.

[2] Jordà, S.: Sonigraphical Instruments: From FMOL to the reacTable*. Proceedings of the 3rd Conference on New Instruments for Musical Expression (NIME 03), Montreal, Canada, 2003.

[3] Jordà, S.: Interactive Music Systems For Everyone: Exploring Visual Feedback As a Way for Creating More Intuitive, Efficient And Learnable Instruments", Proceedings of the Stockholm Music Acoustics Conference (SMAC03), Stockholm, Sweden, 2003.

[4] Kaltenbrunner, M., O'Modrain, M.S., Costanza, E.: Object Design for Tangible Musical Interface: Proceedings of the Cost287-ConGAS Symposium on Gesture Interfaces for Multimedia Systems, Leeds, UK 2004.

[5] Wright, M., Freed, A., Momeni A.: OpenSound Control: State of the Art 2003. Proceedings of the 3rd Conference on New Instruments for Musical Expression (NIME 03), Montreal, Canada, 2003.

[6] Costanza, E., Shelley, S. B., Robinson, J.: D-touch: A Consumer-Grade Tangible Interface Module and Musical Applications", Proceedings of Conference on Human-Computer Interaction (HCI03), Bath, UK, 2003.

[7] Puckette, M.: PD - Pure Data, http://www.crca.ucsd.edu/~msp/Pd_documentation/

[8] Puckette, M.: Pure Data. Proceedings of the International Computer Music Conference, 1996. San Francisco: International Computer Music Association.

[9] Weinberg G., Gan S.: The Squeezables: Toward an Expressive and Interdependent Multi-player Musical Instrument. Computer Music Journal. Vol.25.2, pp 37-45, 2002.

[10] Singer, E.: Sonic Banana: A Novel Bend-Sensor-Based MIDI Controller. Proceedings of the 3rd Conference on New Instruments for Musical Expression (NIME 03), Montreal, Canada, 2003.

[11] Berry, R., Makino, M. , Hikawa, N. and Suzuki, M.: The Augmented Composer Project: The Music Table. Proceedings of the 2003 International Symposium on Mixed and Augmented Reality," Tokyo, Japan, 2003.

[12] Ishii, H and Ullmer, B.: Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms, Proceedings of CHI 97 Conference on Human Factors in Computing systems, Atlanta, Georgia USA, 22-27 March 1997.

[13] SmallFish homepage: http://hosting.zkm.de/wmuench/small_fish

[14] Blaine, T., Perkis, T.: Jam-O-Drum, A Study in Interaction Design, Proceedings of the ACM DIS 2000 Conference, ACM Press, NY, August 2000.

[15] Blaine, T. and Forlines, C.: Jam-O-World: Evolution of the Jam-O-drum Multi-player Musical Controller into the Jam-O-Whirl Gaming Interface, Proceedings of the 2002 Conference on New Interfaces for Musical Expression (NIME-02), Dublin, Ireland, 2002.

[16] Paradiso, J. and Hsiao, K., Musical Trinkets: New Pieces to Play, SIGGRAPH 2000 Conference Abstracts and Applications, ACM Press, NY, July 2000, p. 90.

[17] Poupyrev, I.: Augmented Groove: Collaborative Jamming in Augmented Reality, ACM SIGGRAPH 2000 Conference Abstracts and Applications, p. 77.

[18] Patten, J., Recht, B. And Ishii, H.: Audiopad: A Tag-based Interface for Musical Performance, Proceedings of the 2002 Conference on New Interfaces for Musical Expression (NIME-02), Dublin, Ireland, 2002.

[19] Mathews, M.: The Technology of Computer Music. MIT Press, 1969.

[20] Chadabe, J.: The Voltage-controlled synthesizer. In John Appleton (ed.), *The development and practice of electronic music*, Prentice-Hall, New Jersey, 1975.

[21] Puckette, M. 1988. The Patcher. *Proceedings of the International Computer Music Conference*, 1988, Computer Music Association.

[22] Levin, G.: Painterly Interfaces for Audiovisual Performance, Master Thesis, Massachusetts Institute of Technology, 2000.