The reacTable* - A new table based computer music instrument

Günter Geiger
Music Technology Group
IUA-UPF Barcelona

October 1, 2004

Abstract

The reacTable* is a computer music instrument that tries to surpass the limitations of current instruments in terms of playability, feedback and music experience. We describe a new interface, consisting of a table and objects, that is used as a controller for sophisticated synthesis algorithms. Another novel concept is "dynamic patching" - building the instrument at the same time as playing it, which introduces additional flexibility in the system. "Dynamic Patching" brings together the two concepts of playing and building an instrument at the same time. The synthesizer for the reacTable* is built with the Pure Data computer music system.

1 Introduction

Through the years we got used to computer music performances where the performers sit behind their laptop computers and the only interaction that we can see is clicking with the mouse or writing something on the keyboard. If we are lucky the performers have faderboxes or some other ready made interface that gets used as a controller and we can vaguely follow the steps they are taking in order to make their performance.

Although very sophisticated synthesis methods and sound generation algorithms can be used in real-time on a computer, the interface to control these algorithms in an efficient and flexible manner is still missing. The reacTable* is an attempt to build such an interface. More than that, it is not only the interface, but it is also the synthesizer behind it and a module for visual feedback, which can be used to explain what is happening acoustically.

This paper explains the where and whys of the reacTable*, and outlines a bit the technical details of its implementation.

1.1 The reacTable* as an instrument

The reacTable* is played by moving objects on a table. Each of these objects represent a processing entity of a synthesizer. We could think of these objects as parts of an analog synthesizer, like an oscillator. The player controls parameters of these sound building blocks by moving the blocks on the table. Rotation of the object changes the frequency of the oscillator for example.

Visual feedback gets projected onto

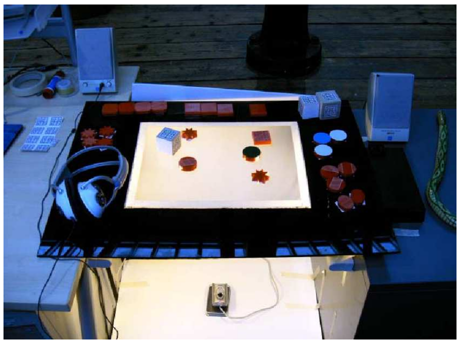

Figure 1: The reacTable* prototype

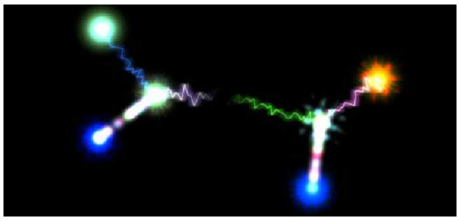

Figure 2: reacTable* visual feedback module

the table, and consists of cords that represent the waveform. This way, if the player puts an oscillator object on the table, the oscillator starts to play and its output is visualized by a waveform that is virtually flowing towards the middle of the table (the sound sink).

If we think of the oscillator as a higher level instrument (like a piano sampler for example) we already have a playable, monophonic instrument.

1.2 Dynamic Patching

The next step of interacting with the table is "dynamic patching". Dynamic patching means the automatic connection and disconnection of units. Unlike traditional patching languages, the patching on the reacTable* is not done by explicit connection of objects but it is governed by connection rules. The main rule is vicinity. Each object gets connected to its closest neighbor.

The automatic connection already takes place when we put one object on the table. The object gets connected to the sound sink (if its a sound generator).

1.3 The reacTable* objects

Generators: Generators are objects that have one or more control inputs, but no sound input and only one sound output. They are called generators because they produce sound on their own. Examples for generators are oscillators, samplers or any other form of sound source such as microphones for example.

Filters: Until now we have only seen one real object type, the sound generator. An oscillator is a sound generator. Other object types are filters. Filters have an sound input and a sound output, unlike generator which only have outputs. Filters do not make sound on their own, but need a sound generator connected to them. Besides the sound input, they have control inputs, just as generators.

Control Generators: These objects are similar to generators, but they generate control data. An example of such an object would be an LFO (Low frequency oscillator). Other more complex examples would be sequencers or chaotic generators. The output of a control generator connects to a control input.

Control Filters: Control filters filter the control data sent by control generators. An example of a control filter could be an object that quantizes frequency data to the MIDI scale.

1.4 The visual feedback

As can be seen in Figure 2, the visual feedback modules projects a representation of the internal state of the synthesizer onto the table. This way the functionality of the objects and their sound output can easily be verified. The black spot in the middle is the sound sink. One can see two different sound flows connected to the sound sink, the sound flow shows the wave form of the currently sounding objects.

Control connections are visualized by dots flowing from control generators to the sound generators and filters. In Figure 2 the two blue spots are control generators. They are both connected to filters and influence the way the filter is sounding.

2 Playing the reacTable*

The goal of the reacTable* is not to make a showcase of what is possible with modern technology, but to try to explore the computer as a new musical instrument. Music instruments have a tradition of thousands of years in human culture, and only recently through the usage of computers we are theoretically capable to build instruments of unbounded complexity and play-ability.

The reality shows though, that most computer music instruments are less playable and less flexible than real instruments. This leads to frustration on the side of the player and the audience. The reacTable* is an attempt to get rid of some of these problems.

2.1 Instrument for novices and professionals

The reacTable* tries to be an instrument that is playable by novices and still challenging for professional players. This means that the table should be playable by a player who sees the instrument for the first time. Just by experimentation and by reacting to the sound and visual feedback given by the system.

At the same time, through dynamic patching, the complexity of the instrument can be very high and it is still challenging for the professional player. The reacTable* is an instrument for virtuosi.

2.2 The collaborative aspect

The reacTable* is designed to be a collaborative instrument as well as a solo instrument. Humans are used to work together on tables, and this can be done in the same way on the reacTable*. The initial version of the table is thought for up to 4 persons. At the same time the table can also be linked via the internet to other tables.

3 The implementation

3.1 The main control unit

The reacTable* is controlled by a software component written in Java which has the task of connecting all the subsystems. There are currently three subsystems of the reactable, the vision engine, the synthesizer, and the graphics engine. The control unit receives input from the vision engine and controls the synthesizer and the graphics engine.

3.2 The vision engine

The vision engine is implemented as a separate program that communicates via OSC (Open Sound Control) with the main control unit. It tracks the positions and directions of the objects and constantly sends its current state to the control unit. The implementation as a stand-alone, platform independent program makes the vision engine useful as a general controller for all synthesizers that implement Open Sound Control.

3.3 The synthesizer

The reacTable* synthesizer is implemented as a Pure Data patch. Pd's message system is used to dynamically instantiate and connect objects together. The objects themselves are implemented as abstractions, so it is easy to change the objects algorithms and build a completely new implementation of the object set based on the reacTable* Pd patch.

3.4 The graphics engine

The graphics engine is a stand-alone program, implemented in OpenGL, that takes care of the visualization of the synthesizer. In order to visualize the correct waveforms, the synthesizer and the graphics engine communicate via the network. In general the graphics engine receives the same information as the synthesizer from the main control unit, and therefor the visualization of the synthesizer state is always up to date.

3.5 Conclusions

We described a new computer music instrument that should enhance the experience of computer music and its possibilities for both, the players and the audience. Building a new computer music instrument is a difficult task, especially if it is not supposed to be a commercial product. We hope that our openness with regards to the implementation, our usage of free software like Pd and open standards such as OSC will further improve the collaboration between researchers in the domain, and eventually lead to better and more sophisticated computer music instruments.