Sounding objects: simulating surface sounds and exploring the mapping possibilities of the reacTable*

Author: Gerda Strobl (Matr. No. 0030841, F750)
Tutors: Günter Geiger, Martin Kaltenbrunner
O. Univ-Prof. Mag. Dipl. Ing. Dr. Robert Höldrich

Project work at the MTG (Music Technology Group) at the Interactive Sonic Systems Group of the Pompeu Fabra University
(Barcelona, Spain)
Institute for Electronic Music (IEM)
(Graz, Austria)

Contents

Abstract

The reacTable* is an electronic music instrument with a tangible interface which is currently developed by the Interactive Sonic Systems Group at the MTG in Barcelona. In this project paper I describe new sounding objects of the reacTable* that simulate surface sounds and explore intuitively the mapping possibilities of the table.
The new objects are implemented using the open source program PD.

Der reacTable* ist ein elektronisches Musikinstrument mit haptischer Benutzeroberfläche, das am MTG in der Interactive Sonic Systems Group in Barcelona entwickelt wird. In dieser Projektarbeit beschreibe ich neue *sounding objects* für den reacTable*, wobei ich versuche Oberflächengeräusche zu simulieren und Objekte zu erzeugen, die auf spielerische Art und Weise die Parameterzuordnung des Tisches integrieren.
Sämtliche neue Objekte wurden mit dem open source Programm PD implementiert.

Chapter 1

The reacTable*

1. The reacTable* a new electronic instrument with tangible interface

1.1 Introduction

The reacTable is an electronic music instrument with a tangible interface which is currently developed by the Interactive Sonic Systems Group at the Music Technology Institute MTG of the Pompeu Fabra University in Barcelona.

The original idea is to create an instrument that appeals to professional musicians as well as to novices that have no musical experience. So the aim is to create a musical system that is intuitive, easy to use and at the same time able to produce more complex and sophisticated sound constructions.

As suggested by its name reacTable* implicates "reactive-table". The table allows the direct manipulation of objects in a synthesis chain. By arranging a set of objects on the table surface the performer constructs and plays the instrument at the same time. Each of the objects has a dedicated function for the generation of sound, modification and control.

While the table itself is equipped with sensors for the identification of the objects the performer need not wear any controllers or devices.

Figure 1.1 reacTable* prototype

1.2 Description of the table

The reacTable itself is based on a rectangular transparent table (a round table is being aimed) which has no sensors, no cables graphics or drawings. The only devices the reacTable* user has, are a collection of about twenty-five plastic objects to generate music.
The sound synthesis engine is implemented using the open-source language Pure Data.
All these components are completely independent and communicate via a simple network protocol. This separation allows execution on various platforms.

1.2.1 Computer Vision

Right in the center at a specific position under the table is a web camera that permanently analyzes the surface of the table and detects type, position and orientation of the objects that are distributed on the tables surface. The web-camera is directly connected to the d-touch computer vision framework [9] which produces the coordinates and types of objects seen by the camera. The d-touch framework is based on the localization and recognition of fiducial markers, namely black and white graphical symbols that can be printed on labels and are put on the bottom of the objects.
Out of each fiducial symbol the computer vision gets the following informations:

- ID
- Position(x,y)
- angle (-180º to +180º)
- 1, which serves for the orientation

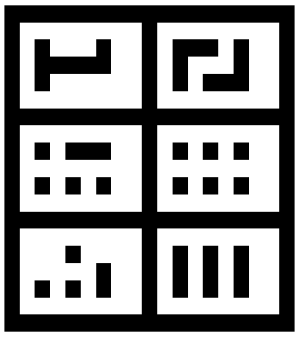

Figure 1.2 example of an fiducial marker
Currently a marker set of 128 different fiducial symbols exists though only a couple of them are used due to the limited table size.

A sensor module tracks the state, the position and the orientation of the any object that is on the table, within the possible radius the camera can get. These raw sensor parameters are passed to the central management component, which interprets the users gestures based on the incoming data, generating a dynamic patch network (that drives the two actual synthesis components for the sonic and the graphical feedback.)

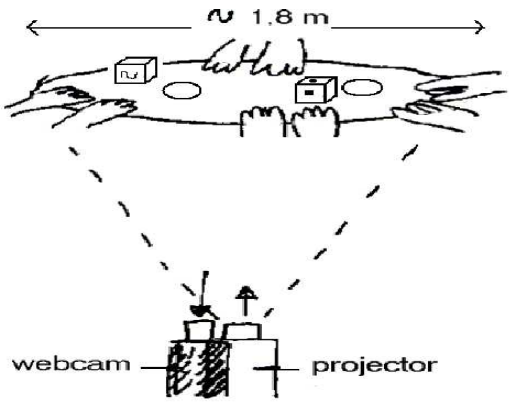

Figure 1.3 reacTable* simplified scheme

1.2.2 Mapping Parameters

Unlike the real table is rectangular there is a centric mapping, that is calculated by the Java algorithm out of the informations that are retrieved by the computer vision. In the middle of the table a point is assumed to be the zero-point.

Whenever an object is recognized eleven mapping parameters are calculated:

1. angle(1); angle of the object, changes when the object is rotated
2. centerdist(2); distance to the center
3. centerangle(3); angle to the positive x-axis
4. nextdist(4); distance to the next object in a synthesis chain
5. nextangle(5); angle to the next object in a synthesis chain
6. movespeed(6); velocity when an object is moved from point1 to point2
7. rotspeed(7); velocity when the angle(rotation) of the object is changed
8. prev0dist(8); distance to first object in a mixing object
9. prev0angle(9); angle to first object in a mixing object
10. prev1dist(10); distance to second object in a mixing object
11. prev1angle(11); angle to second object in a mixing object

1.2.3 Visual Feedback

The graphical feedback is being produced by a video projector that is being fixed under the table which draws a dynamic and interactive surface on the table.
The projection follows the objects on the table wrapping them with auras and showing their actual connections. The projection is not logic, which means, that no numerical or textual information is shown, except of the control or audio connections between the objects.
When a control flow is established between two objects a thick straight line is being drawn between them, showing by means of dynamics the flux direction. While control flow lines are straight lines, audio lines are organic and complex as shown in figure 1.4

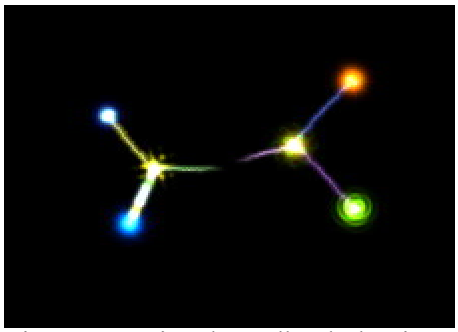

Figure 1.4 Visual Feedback that is projected on the table.

1.3 Objects

All the sound generating objects are three dimensional, made of plastic, plain and passive which means that they do not have any embedded electronics. They react as physical tangible representation of the various virtual synthesis components.
All the objects that can be used have a haptic dimension such as shape, size and material to create a suitable haptic encoding scheme for the various abstract object types and their variations, in order to allow rapid and accurate object identification by simply grasping them with the hand.
The objects can be generally categorized into five different groups:
Sound Generators, Audio Filters, Controller, Audio Processors and Mixers:
So far the label size, and thus the object size is constrained by the system resolution. Actually this depends not only on the resolution of the camera but more obviously on the available processing power.

1.4 Connection Rules

Dynamic Patching does not require the user to explicitly connect the objects because there is a predefined set of rules that automatically connect and disconnect objects. In contrast to the idea of PD, there is no distinction between editing and runtime mode, both modes are carried out at the same time on the table
All the objects are connected by certain proximity laws. The algorithm consists of a tree like structure, testing the availability and compatibility of the closest port (audio or control), and searching always the next closest connection.
Moving an object on the table automatically changes its relation with the other objects, as an object always looks for the next available object. If upon start-up a user activates an object that does not sound (i.e. a control object) the closest audio object is automatically linked to it.
Objects remain not active until they are positioned on the table, picking them up acts as a mute, whilst putting them down on the table reactivates them.

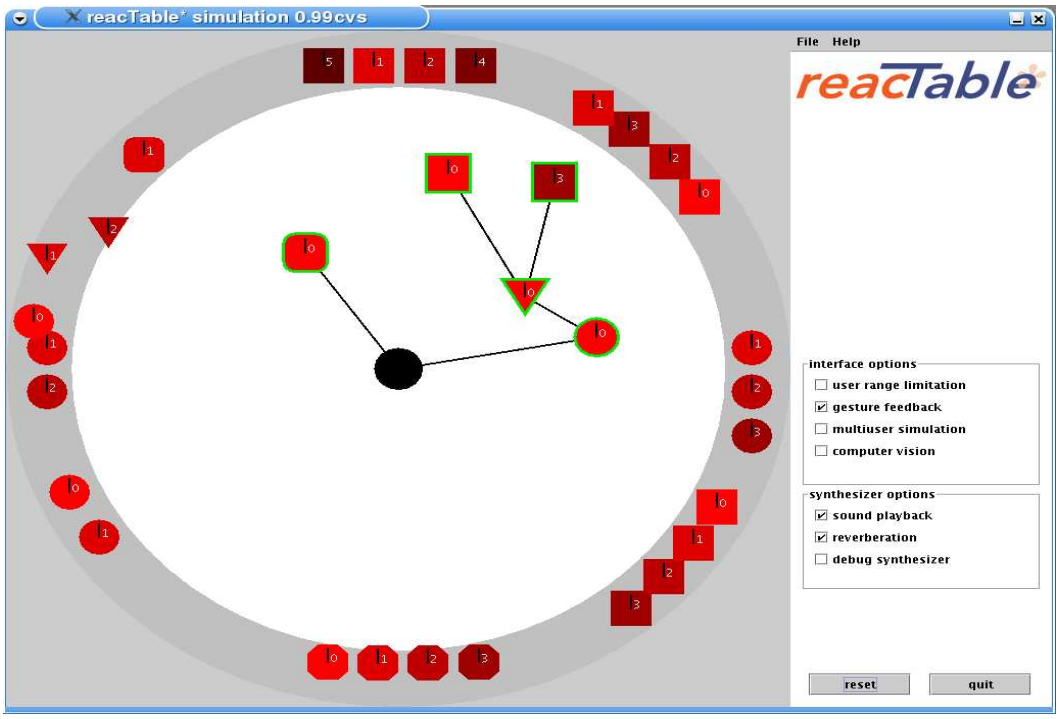

Figure 1.5 reacTable* simulation

Chapter 2

Synthesis Engine

2. Implementation of the audio synthesis in PD

2.1 Connection to the Java engine and mapping

As mentioned before there are eleven position parameters that the PD synthesis engine permanently receives from every activated object. To communicate with the Java engine three different text files exist that are read by the Java algorithm.

2.1.1 Textfiles

default.typ defines the basic structure of an object concerning the sound in/outputs, control in/outputs, side, and shape.
default.rts specifies the start-up position of the objects on the computer simulation.
The eleven incoming parameters serve as input parameters for all the synthesis objects. Within the **default.map** the coordinates belonging to every specific object are defined which are received by the x_mapping.pd file that is available in every pd-object file.
In this file the parameters can be scaled such that they can be used for the objects, defining offset, threshold, range and whether their scalar range is logarithmic or linear.

2.2 Main.pd

The central unit that distributes the incoming *netreceive* informations is included in the main.pd patch. Whenever objects appear on the table all the eleven mapping parameters are sent to the *netreceive* -object that passes them to a router, that sends the parameters in the order they arrive to the sub-patches that are connected.

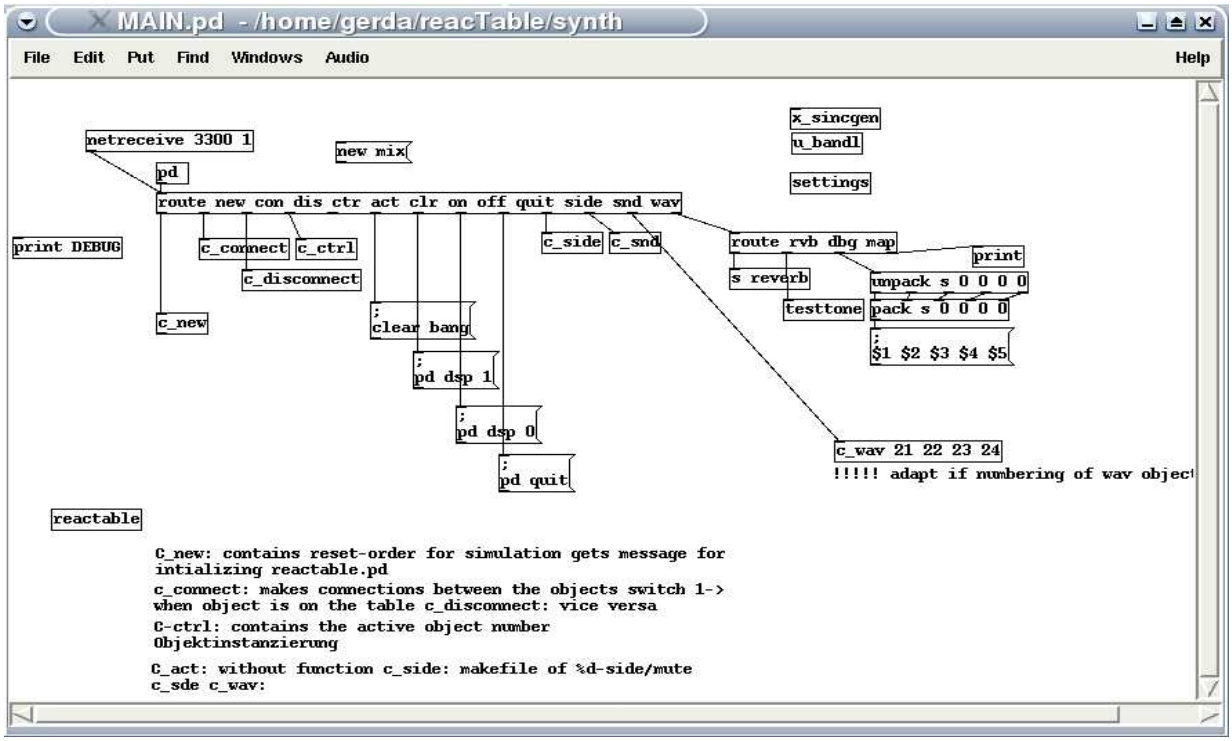

Figure 2.1 main.pd

2.2.1 Subpatches of main.pd

c_new:

Contains the order for initializing the reactable.pd file that contains all the existing objects and receives the information whenever the simulation is being reset.

c_connect:

Sends a switch-message whenever an object is on the table and sets up the connections to the objects and to the final output object; a disconnect patch exists as well.

The fade in/out - files that are included in every file are activated.

c_ctrl:

Sends out the current control number of the active objects.

Additionally messages for clearing the table and switching on/off the dsp unit are distributed.

c_side:

Some objects e. g. the cubic sampler are singular objects but have more sides to be used, that is why the c_side patch sends the side information to the sub-patches which have special side-receives.

c_snd:

The mute - files that are included in every file are activated.

2.3 reactable.pd

Whenever the reacTable* is activated the reactable.pd file is dynamically allocated. It contains all the available sounding objects that can be used on the table. In the current version 33 objects are defined; object number 34 is the outlet, that combines the audible output of all the active objects, contains the reverberation and sends all the informations to the digital audio converter of the sound card. This objects is included in the reactable.pd file but has no representatin neither on th table nor on the simulation.

Generally the objects can be categorized into five different groups:
Sound Generators, Audio Filters, Controller, Audio Processors and Mixers:

2.3.1 Sound Generators

The sound generating group contains all the oscillators, the samplers, the granular objects and the wavetable oscillators.

As I have given the oscillator object special importance ten different "oscillators" are available although only six can be used.

The old oscillator collection consists of (osc0-osc5)

x_noise old (pure white noise); x_saw; x_pulse

The new objects(x_blin, x_sandpaper, x_scratch, x_bell, x_toc and x_quiek) are described in detail in the chapter 3.

The sampler (smp17-smp23) contains six different sound samples, that can be changed according to the use of the reacTable. Two cubic sample objects exist for mixing different samples together.

The wavetable oscillators (wav 23-24) reads out a predefined cosine wavetable. Actually their intentional idea is reading out a waveform that has been drawn by a table user beside the object by the hand or a certain pencil. Up to now this only works in the simulation when drawing a waveform with the mouse as the computer vision is only able to recognize the fiducial markers.

The granulator object (gra 25-28) implements granular synthesis. Four different sound samples serve as input.

2.3.2 Mixers

The mixing (mix31, fm32 and am33)objects are described in detail at point 4.

2.3.3 Audio Filters

There are three audio filters that act as band pass filters. When connected to an audio generator they change the frequency of the object by using the their rotation as center frequency (flt6-8).

2.3.4 Audio Processors

The effect group belongs to the group of the Audio Processors depending on the selected side of the object four effects are provided: x_karplus (plugged string effect); x_phasor; x_chorus; x_delay (eff 9-12).
Two envelope objects which build envelopes, that depend on mapping values, for other objects whenever they are connected and moved (env29-30).

2.3.5 Controllers

After all there are four controllers (lfo 13-16) that have only control outputs. Depending on the object side they send out control values being part of a sine, saw or noise signal. Their angle to the object they are connected to defines the velocity of their sampling rate. In addition *henon* a chaotic noise generator (pd external for the reacTable*) is connected distributing the control values at random.

Chapter 3

Sounding Objects

3. Sounding Objects: exploring the mapping possibilities of the reacTable*

3.1 Introduction

Up to the now the oscillator group, has been very technically orientated; Sawtooth, white noise, sine et cetera. Some of the objects offered a haptic surface which made me think of making surface sounds, simulating the friction of objects as predetermined by their haptic face.

The Sounding Object Project [8] SOb aimed at developing sound models that are responsive to physical interactions and are easily matched to physical objects. Sound and control models were developed after the phenomenological and psychophysical characterization of a restricted class of sound events. The results of this research were demonstrated by means of a dynamic sound library.
By use of physical modeling bouncing balls, impact sounds, and struck materials were simulated in PD. Trying the patches that are available on the SOb web page gave me an insight into creating sounding surface scenarios. Although the sound examples modeled certain sounding aspects amazingly real, taking ideas from that project did not support me by creating objects for the reacTable*.

Keeping in mind that the reacTable* intends to be an electronic instrument that aims to be intuitive I decided to create objects that correspond to their surface (sandpaper making sandpaper sounds) that make interesting friction sounds that might be possible on the actual table surface (squeaking, scratching, rasping). Herby real sounds served as an orientation but realistic simulation was not the perfect goal.

The first reacTable* objects did not change their sound characteristcs or pitch when they were moved across the table.
Thinking of using a normal instrument where sound production is controlled by moving the hand across a surface, (Glissandi, a run up and down a piano, vibrato et cetera) I decided to use the various mapping parameters for creating appealing objects dealing with the local parameters (bell, blin, toc).

3.2 x_blin

The blin-object mixing object combines twelve oscillators that change their input at random, whenever the object passes a certain mapping value which results in a jingling sound that changes whenever the object is relocated.
The blin object uses the distance to the center as the main input. Within the resource file I arbitrarily chose the scaling of the center-distance to be between 10 and 60 . The incoming values are tested whether they fit the function *select*, whenever select sends out a positive trigger the appropriate random generator passes on a number.
For making a harmonic order I decided to start with a baseline where the center distance has its biggest values.
Collateral counters are included. Two of them sweep up, two sweep down, starting from the input value. Whenever the object passes again their defined mapping value a new sweep is started.

If a controller is linked the incoming values are multiplied by a scaling factor as they are quite low and added into an oscillator. Additionally the amplitude of the incoming control frequency is increased to give the impression of a bubbling mid frequency line.

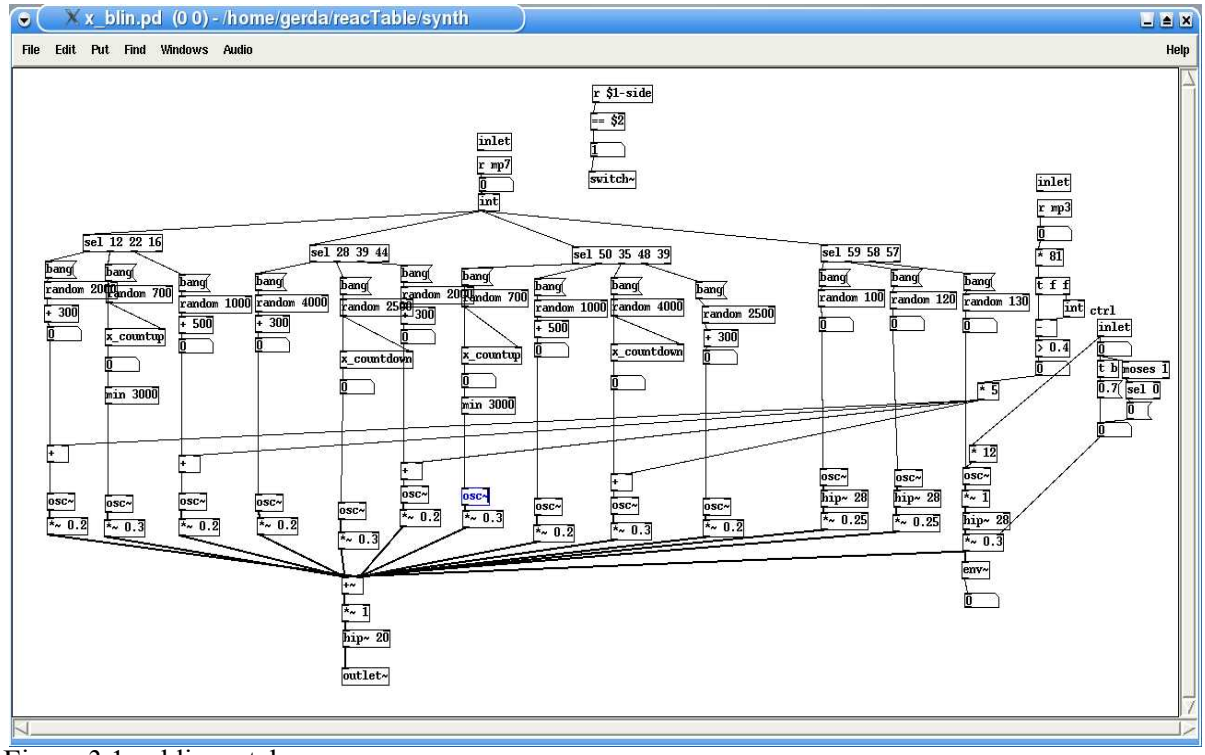

Figure 3.1 x_blin-patch

3.3 x_bell

As the name implies this object sounds like a bell. The bell-like sound is produced using frequency modulation. The rotation of the object serves as the carrier frequency. If either the carrier frequency f_c or the modulation frequency f_m is an irrational number it results in an inharmonic spectrum. Using the ratio of $f_c/f_m=\text{sqrt}(2)$ [1] produces a bell-like sound.

As this is an object that has no further audio input I simply multiplied the carrier frequency by the mentioned ratio to simulate the modulation frequency. The modulation index is provided by the center-angle which has a range of 300.

The specific bell sound is of finite duration, that's why the envelope (short attack, long decay, no sustain and no release) appears whenever the object is moved to give the user the impression that whenever he changes the position of the bell object, the bell is being struck.

The linkage of the controller effects the change of the modulation frequency. Whenever connected the specific bell sound is canceled.

3.4 x_scratch

Produces scratching surface sounds whenever the object is moved across the table. Rotating the objects results in a change of timbre whereas changing the center angle position of the object produces the rubbing scratching effect.
The patch consists of two different oscillators that are multiplied. The frequency of the oscillators that somehow serves as a carrier is defined by the centerdistance within the range of [50-150 Hz]. If the right inlet of the *osc~* object is changed the phase is reset. Like I have already mentioned above the rotation of the object changes the timbre. Actually I use the rotation as input for a metronome, that resets the phase of the *osc~1*. A slow metro output produces a high frequency output that is similar to plugging a string instrument below the bridge.
The frequency of the second oscillator varies only between [0 - 4 Hz] and serves as the modulation frequency. The phase of *osc~2* is also reset as well by using the second metronome. In that case the center-angle scaled by 1000 is used as metronome input. As the range of the input values is much higher, great number changes, corresponding to movements across the table, effect the rubbing, scratching effect.
If the scratch object is not moved, just the pulsing sound effecting the phases of two oscillators can be heard.
Originally I intended to mute this object whenever it is not moved, but by chance I combined the scratch object with the karplus-strong effect which provides an interesting sound-combination.

3.6 x_toc

Starting with the idea of walking along a batten fence whilst letting glide a stick along , producing a well known rhythmic sound depending on the material of the fence and the walking speed, or taking the sound of a güiro as an example (an untuned percussive instrument from Central America, which is made of carved gourd that has a ridged surface) which has a very specific rasping sound, made me think of simulating the ridged surface with the reacTable* and using the object passing across the surface as an exitator.

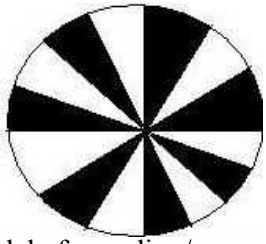

Figure 3.2, model of sounding/non-soudning areas

Using the center angle parameters and circle-form of the simulation interface I divided the circle into 128 sounding and non sounding sections. Whenever the object passes a non sounding section a zero value is sent to the output to reset the amplitude. Depending on the number of sections and of the velocity a rhythmic pattern can be created.
Keeping at the idea of the two proposed sound examples I decided to insert a noised signal with a specific envelope, that is changed by mapping parameters, into a *vcf~*(voltage controlled bandpass)-object. The center frequency that is as well combined by a specific envelope is defined by the rotation of the object. Additionally the Q-factor is defined by the movespeed parameter; the higher the movespeed the softer is the rasping sound.

3.7 x_sandpaper

Within the collection of the real reacTable* objects there has already been an object with a haptic sandpaper surface before I started working. In fact the timbral properties of the object sound did not correspond at all to the rough surface the user touches when using the object.
That is why I decided to implement a similar sound by the use of noise generation. The real sandpaper sound, that is offering a great variety, served as an orientation, but complete realistic simulation was not my goal as the reacTable* is a musical instrument.
As the basic input I use simple white noise that is processed by two high pass filters and one bandpass filter to get high noisy components. Then the filtered white noise is multiplied with a saw tooth signal having a high amplitude. A further sawtooth created by a phaser signal using the movespeed-parameters as input frequency is modulating the first noisy sawtooth signal. The control inlet is also connected to the phaser inlet to change the scraping sound.
Once again I added a small loop that divides the surface into sounding and non sounding areas to sharpen up the impression of having a resisting surface.

3.7 x_quiek

So far the reacTable* prototype that has been built is having a white acrylic glass table top such that a self evident thing was to produce a sound that deals with that surface. The most interesting glass sound worked out when rubbing or polishing glass with a wet hand or
with newspaper.
To create a squeaking sound I decided to use granular synthesis as I regarded it practical
taking the tables speed parameter to change the granulation length and density.
The first part of the patch takes advantage of the already existing m_granular.pd-patch, that divides the sound into a series of grains derived from various wave files; though using only one "household"-sound-file and connecting other input parameters. The grain's pitch is defined by the movespeed-value and the centerdist (when the object has to sound but is not moved). The rotation of the object defines the grain size. Like in the granular objects the centerangle defines the offset but within this object I restricted the incoming values to a small range which generates a more favored sound.
The output of the m-granular patch is high-pass filtered, multiplied by a fm modulation patch (x_brumm) and added to the output of an other granular object: x_Granquiek.
Granquiek uses the same sound-file and takes the output of the *soundfiler* (sample-length) as input for a random generator that selects a random grain position. The rotation parameters are used as grain length, they are divided, passed through a random generator to get their duration; finally they are again reconverted to the sample-length; the new sample-length parameters and the random granular position are packed into a *tabplay~* object.

In order to simulate the friction of a wet cloth rubbing over the surface, the object may sound only when it is moved, that is why an additional object "x_ObjOnOff" can be added, that mutes the object whenever it is in a passive state.

I also tried using other sound-files as input for the granular synthesis; piano files turned out to make as well very interesting sounds but get side-tracked from the original idea of creating a surface sound.

Chapter 4

Mixing Objects

Audio Synthesis objects using two audio inlets

4.1 Introduction

So far there was no possibility to combine mix or modulate objects sending out audio signals at the reacTable. As sound synthesis is an essential issue in computer music to generate interesting music we decided on making three new objects with two audio inlets, one audio outlet and as in all the other objects one inlet for the lfo-controller. If only one audio inlet is connected theses objects act as troughs.
I finally decided on implementing a normal adding mixer, an amplitude modulation object and a phase modulation object because these three methods of mixing two audio signals worked out the best for the existing reacTable* objects.

4.2 Mixer

The mixer combines two audio objects and sums them up. By using the parameters prevdist1 and prevdist0 explained in 1.2.2 the amplitude of the incoming signals is changed. The closer an object is to the mixer the lower is the volume. Rotating the mixer changes the main volume.
If a lfo-controller is connected its values influence the stereo panning by changing the angle, which results in a quickly wandering sound depending on the rapid change of the control values.

4.3 Phase Modulation

The mixing objects have by default two inlets receiving just audio values. I initially wanted to create a fm-modulation object, unfortunately this did not work as the fm modulation (one signal is modulating the other) concept is contrary to the idea of mixing two audio signals into one.
That's why the only real fm-implementation can be found in the x_bell object in the oscillators group described above.
As well in the AM-, as in the PM object-the connection rule implies that the first audio object is automatically connected to the left inlet of the mixing object and I defined the left inlet by default to be the frequency inlet for the carrier.
For changing carrier and modulator objects just have to be disconnected and then be again connected vice versa.
Starting with the basic idea of carrier and modulator the signal of the carrier is being written into a delay line; the amplitude of the modulator frequency is changed by the rotation of the fm object;
the modulator is connected to a *vd~* object (reads a signal from a delay-line and takes the inputs signal as delay time). Thus the modulator apparently changes the phase of the carrier producing interesting sounding effects especially when one of the inputs is a sampler.

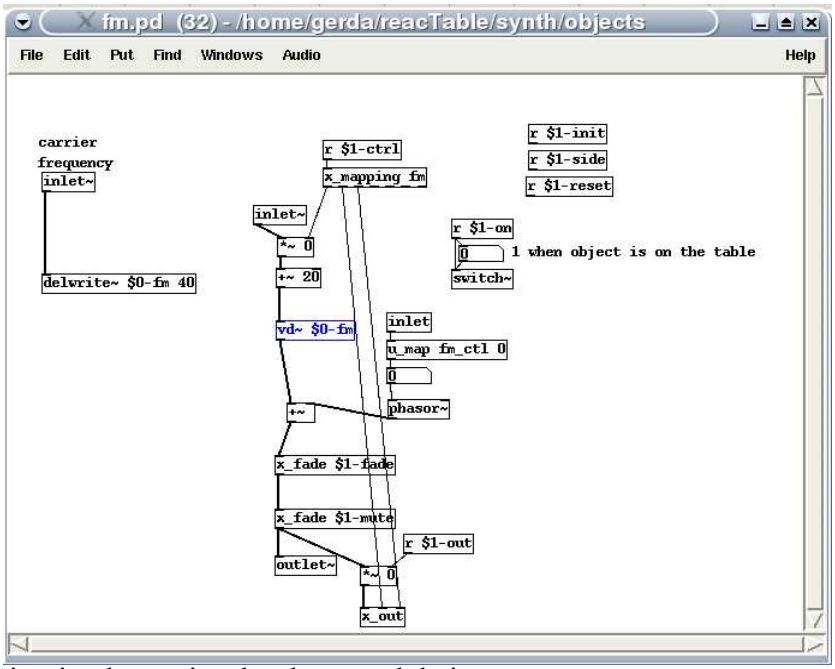

Figure 4.1 fm object implementing the phase modulation

4.4 AM

I implemented the classical AM in which the modulator, the right inlet is restricted to be a unipolar signal having the range between 0 and 1. The left inlet, the carrier remains unchanged bipolar.
Rotating the objects changes directly the modulation index.
Like in the mixer if a lfo-controller is connected, its values influence the stereo panning by changing the angle, which results in a quickly wandering sound depending on the rapid change of the control values.

Chapter 5

Conclusions

5.1 State of the art

In June for the NIME conference in Hamamatsu (Japan) the first real reacTable* prototype was built having the table size of 50 * 35[cm], the visible size for the computer vision is about standard paper size (DIN A4).

Figure 5.1 reacTable* prototype for Hamamatsu

In contrast to the computer simulation that is having a high resolution, the frame rate we have with the prototype at a resolution of 320 by 240 pixels is producing about 25 frames per second. That means very precise or rapid object movements can not be calculated, the speed parameters are very inaccurate and lifting an object rapidly from the surface produces an undesired delayed mute. Increasing the image resolution over 640 by 400 pixels would result in an unacceptable temporal resolution demanding a very high processor cache.
One further side effect of the computer vision is poor lighting or motion blur, occurring whenever objects are moved to quickly.

There is also the idea of integrating the hand as a reacTable* object. In the computer simulation drawing a line with the mouse that crosses a connection deactivates the connection until it is touched again. However tracking the hand as an object is still not state of the art of the computer vision. Also as previously explained the lfo-oscillator allows the manipulation of its waveform by painting a waveform aside with the "hand".

5.2 Usability discussion

The reacTable* intents to be an instrument as well for novices as for professionals. Up to now the table has been used by both when the table was presented to the public. What has to be mentioned that whenever people were trying the table someone knowing the functions was standing aside.
Right now the table includes a lot of technical objects, that do only sound when they are connected to an audio generator. For me this raises the question, which objects make sense of being used when presenting the table, using it as an instrument for musicians or when it is exhibited in the context of a sound installation.
As the objects on the table and their functions can be easily changed I suggest adapting the objects to the context in which the reacTable* is used to satisfy the demands of the user.
I am sure a beginner wont be disappointed having no envelope or filter for use as trying out all the sounds should be the first encounter with the table.
In contrast to a novice a skilled person having knowledge of computer music certainly wants to have more controlling objects for producing high sophisticated pieces.

5.3 Acknowledgments

I would like to thank Günter Geiger for offering me that much personal support and to Martin Kaltenbrunner who spent a lot of time explaining me all the technical context of the reacTable*. After all I want to thank to the Sergi Jorda, the reacTable* philosopher and the MTG providing me the possibility of doing my project work at this place and Robert Höldrich supporting me to go abroad.

Appendix

A.1 Contents of the accompanying CD Rom

- An executable version of the reacTable* simulation for Linux
- All the new sounding objects are implemented within the oscillator group or can be found directly in the synth folder:
    - x_bell.pd
    - x_blin.pd
    - x_quiek
    - x_sandpaper.pd
    - x_scratch.pd
    - x_toc.pd
- The mixing object can directly be found in the objects folder.
    - mix.pd
    - am.pd
    - fm.pd
- A demo video of the running table made for the NIME conference in Hamamatsu.
- Further photos showing the use of the reacTable* prototype

References

[1]Charles Dodge, Thomas A. Jerse "Computer Music Synthesis, Composition, and Performance" 1997
Synthesis using Distortion Techniques pp 115-135
Granulation of sampled sound pp 269-271

[2] Martin Kaltenbrunner, Guenther Geiger and Sergi Jordà "Dynamic Patches for Live Musical Performance" in *Proc. of the 4rd Conference on new Instruments for Musical Expression (NIME 04)* Hamamatsu, Japan 2004

[3] Martin Kaltenbrunner, Sile O'Modhrain, Enrico Constanza "Object Design for Tangible Musical Interface" in *Proc. of the Cost287 -ConGAS Symposium on Gesture Interfaces for Multimedia Systems,* Leed UK 2004

[4] Sergi Jordá "Sonigraphical Instruments: From FMOL to the reacTable*" in *Proc. of the 3rd Conference on new Instruments for Musical Expression (NIME 03)* , Montreal, Canada 2003

[5] Sergi Jordá "Interactive Music Systems for Everyone: Exploring visual feedback as a way for creating more intuitive efficient and learnable instruments" in *Proc. of the Stockholm Music Acoustics Conference* , Stockholm 2003

[6] Curtis Road "The Computer Music Tutorial", 1996
Modulation Synthesis pp 211-315

[7] Matthias Rath, Nicola Bernadini, Federico Fontana, Davide Rocchesso "An introductory catalogue of computer-synthesized contact sounds, in real-time" in *Proc. XIV Colloquium on musical Informatics (CIM)* , Firenze 2003

[8] Davide Rocchesso, Federico Fontana, editors "The sounding objects" 2003

[9] Simon Shelley, John Robinson, Enrico Constanza "Introducing audio d-touch: A tangible user interface for music composition and performance" Proc. on the 6th Int. Conference on Digital Audio Effects (DAFX-03), London 2003