Texture: a Granular Synthesizer for Real-Time Sound Generation

GIORGIO NOTTOLI[1], GIOVANNI COSTANTINI[1,2], ANDREA ANGELINI[1], MASSIMILIANO TODISCO[1], DANIELE CASALI[1]

[1]Department of Electronic Engineering
University of Rome “Tor Vergata”
Via del Politecnico, 1 - 00133 Rome
ITALY

[2]Institute of Acoustics and Sensors “Orso Mario Corbino”
Via del Fosso del Cavaliere, 100 - 00133 Rome
ITALY

massimiliano.todisco@uniroma2.it

Abstract: - Sound synthesis is a subject where last development of electronics had made a significant boost. Since it’s beginning, it developed on two main branches: on one side it tried to imitate other instruments, trying to re-create sounds that already exist. On the other side, it followed an aim that we can consider someway opposite: producing new sounds, exploring new possibilities, and allowing composers to follow new paradigms in musical composition. Imitating other instruments can be useful for various reasons: the electronic version of an instrument has often very lower cost with respect to the original one, and also lower weight. Besides, it can have some advantages, as the capability to be played by a computer or to use headphones. Producing new sounds, on the other hand, means often better fitting than any acoustic instrument with the idea of a musician. The issue, in this case, is to give the musician the opportunity to easily control all necessary parameters, in order to obtain the desired result. So, while on the first branch we can easily say that the better instruments are the ones that imitate more closely real instruments, on the second branch the variety of produced timbres is important, but it is almost useless if it is not accompanied by a tool to control a huge set of parameters in an efficient way. In this paper, we focused on the production of new sounds, and we present a system called Texture, that generate sounds in real-time. The current version of Texture is available both on Windows and on OSX operating systems, both as a Virtual Studio Technology (VST) and as Audio Units (AU). The system is based on the Granular Synthesis, which is a method that produces complex sounds by mixing together simple elements called “grains”, but it extend the classical method with some new features that bring more richness and variety to the sound. The software comes with a graphical interface and applications that allow to control the synthesis parameters in an effective way, and that give the musician the opportunity to add expression to the sound. This goal is reached by means of neural networks.

Key-Words: - Real-time systems, sound synthesis, granular synthesis, audio plugins, VST, AU.

1 Introduction

Electronic music has traditionally made use of analog and, later, digital sound synthesizers which was based on DSPs, because the power required for real time sound synthesis at a reasonable quality was initially not available on a computer [1]. In fact, to obtain a real-time stereo signal at a sampling rate of 44.100 Hz, which is the standard quality of a CD, we must be able to calculate and output the value of a sample in a maximum time of 11 microseconds. This goal was difficult to reach on a general-purpose processor, both for limitations related to the power, and because ordinary operating systems which run on PCs are not designed to work for real-time operations, i.e. it is not guaranteed that a given process receive the control of the machine within a so short time interval (typical task switch times for a late 1980’s processor are roughly 20 microseconds). This forces real-time audio applications to use large buffers, which produces a latency that is often unacceptable in a live execution. Audio boards added some latency as well. All these issues oriented sound engineers to the use of DSPs for audio synthesis and processing systems [2,3].

In years 90’s some notable hardware systems for real-time sound synthesis and elaboration have been developed.

One of these is MARS (Musical Audio Research Station), produced by IRIS, the research institute owned by the Italian Bontempi-Farfisa group [4,5]. The system included a pipelined architecture, fully programmable, 40 MHz system clock, and a sampling rate of 39.0625 KHz up to 1 MHz. This DSP power allowed it to implement in real-time 128 second order filters, 256 oscillators, 128 delay lines and 2 independent 2048 points FFT.

Surely, a relevant system of years 90's is the Kyma [6], realized by the Symbolic Sound Corporation. It is a graphical modular software sound design environment accelerated by the software-reconfigurable Capybara multi-processor sound computation engine. The Capybara is a powerful sound computation engine designed to work in conjunction with the Kyma sound design environment its innovative hardware and software offerings.

Finally we can mention SAIPH [7], a system developed in the University of Rome "Tor Vergata" and intended for research in the field of real-time sound synthesis and elaboration. It is made by two different subsystems: Betel Orionis for the sound synthesis [7], and Rigel for the sound processing and spatialization [8]. The first one is based on the dedicated DSP Orion, while the second one is based on two fixed point DSPs of the 56300 family by Motorola: the 56301 DSP and the 56302 DSP. Saiph is intended for live performances in concert halls and it is interfaced with a personal computer and MIDI controls that allow the composer or performer to interact with it.

In years 2000s the required computer power started to be available on PCs, and software modules for real-time audio synthesis overcame DSP based systems. Advantages of a software system are not limited to economic reasons, but also include a higher versatility, programmability and easier design.

In 1996 Steinberg developed Virtual Studio Technology (VST) [9], a software interface that integrates software audio synthesizer and effect plugins. VST plugins became the standard for audio software modules, and generally run within a digital audio workstation (DAW), to provide additional functionality. For Mac OS X, Audio Units (AU) plugins are also used.

A software that we have to mention is Max/MSP [10], developed in the mid-1980s as the *Patcher* editor for the Macintosh. It was later ported to the IRCAM Signal Processing Workstation for the NeXT (and later SGI and Linux), and then licensed to the Opcode Systems. Currently it is distributed by Zicarelli's company Cycling '74. The software offers a visual environment to design real-time audio applications, called *patches*.

The system presented in this paper is an evolution of the algorithms implemented in Betel Orionis, here developed on a software system. It integrates different kinds of graphical interfaces, and capabilities to interact with other software. The current version is available for Windows and OSX operating systems, both as VST and AU modules for Mac systems. An up to date version of Texture can be downloaded from [11].

This paper is organized in four sections: in the first one, we will discuss the synthesis method used by this software, which is the Granular Synthesis, with some historical considerations; in the second one, we present our system, which is called Texture; in the third section, we describe some real-time applications where we use the software Max/MSP and neural network based algorithms to control the complex set of parameters for the sound generation. Two kinds of applications are described: in the first one the user moves a point on the screen and the neural networks map the two-dimensional space given by the position of the mouse to the *n*-dimensional space of parameters for sound synthesis; in the second application, a glove with bend sensors is used to get the position of the hand, and the neural network outputs a set of parameters for sound synthesis. Even in this case, the task is done by means of a mapping from *m*-dimensional space to *n*-dimensional space. Both applications are designed to give the musician the possibility to put expressivity in the sound. Finally, the last section is dedicated to the conclusion.

2 Granular Synthesis

"The grain is a particularly apt and flexible representation for musical sound because it combines time-domain information (starting time, duration, envelope shape, waveform shape) with frequency domain information (the frequency of the waveform within the grain)" (Curtis Roads) [1]

Granular Synthesis [12] is a sound synthesis method that allows to generate complex tones from the combination and the mixing of simple micro elements called grains.

Usually each grain has a time length range between 10 and 50 ms, and can be reproduced with different speed, phase and volume; it is an acoustic event with its own envelope and a very small duration, near the limit of human auditory perception.

Increasing the playback rate from low to higher speeds, the sound texture created will turn from a noisy cloud of micro sounds to a distinct and organic sharp tone; depending on the waveforms and the envelopes of the grains it will be possible to generate many different kind of timbres, employing fixed waveforms (as sine or saw), dynamic ones (as FM synthesis) or sampled waveforms.

The need to work with thousand of micro elements requires a higher computational cost than other synthesis methods historically more popular, like subtractive synthesis, being one of the reason why artists started to discover and explore granular synthesis only in the last times.

2.1 Historical Notes

Even if granular synthesis can be considered an innovative method whose evolution has hugely grown with the use of computers and modern recording techniques, its theoretical background can be found in the human perception of the sound.
Dutch scientist Isaac Beeckman (1538-1637) theorized that sound propagates through expansion and contraction of air particles density, considering sound formed by several globules and thinking about a corpuscular theory of sound.
Since 1900, with quantum Mechanics studies of Max Planck and a discontinuous theorization of the light, similar theories were applied to sound from Norbert Wiener, who noticed a relationship between pitch and time; he observed that a note of 20Hz played for less than 1/20th of a second would produce no sound.
In 1946 Nobel-prize physicist Dennis Gabor presented the article ‘Theory of Communication’, which described a system to reproduce granular synthesis. He theorized the grain as a rectangular area in the domain of time and frequency, such that decreasing the sound duration will produce an increasing in the frequency domain.
One year later he wrote ‘Acoustical Quanta and the Theory of Hearing’, discussing some issues with the Fourier theory applied to common sounds with variable frequency like the one of a siren; moreover the Fourier theory needs sine waves with infinite duration to generate sounds. He suggested to apply quantum physics methods to sound signal, reducing the Fourier method into cells in a mathematical context.
He also built some machines with only mechanical parts to demonstrate what he called kinematical method of frequency conversion, in which a sound film ran into a film projector at constant speed and the light accessing to a photocell through slit was converted to sound signal.
On the basis of another Gabor machine, that could record and erase tape in a loop, the German company Springer built and commercialized the Tempophon, used in 1963 by Herbert Eimert for the electronic music composition ‘Epitaph für Aikichi Kuboyama’.
In 1960 Xenakis followed the Gabor study integrating it with stochastic methods [13]; he theorized that every sound is formed by a enormous number of particles or elementary grains, organized in what he called ‘Screens’ sequences.

2.2 Grain - Sound Content and Envelope

The grain sound content can be generated by every kind of sound wave, windowed in time domain.

- Tapped Delay Line Granular Synthesis, is a method in which samples from a realtime stream are stored through delay lines. Each grain will later be read by delay line with its own time and playback rate
- Stored Sample Granular Synthesis, creates each grain reading values from a wavetable
- Synthetic Grain Granular Synthesis, generates totally synthetic sound textures, using standard synthesis methods like oscillators or FM synthesis

Each grain envelope can be controlled in duration and amplitude by parameters like attack, decay, sustain and release; moreover the grain should be windowed in time domain inserting a fade in and a fade out in order to avoid clicks when moving from a grain to another.
Typical grain time duration is between 10 and 50ms; in fact shorter grain will interfere with the comprehensibility of the sound content creating a texture with a strong noise component, on the contrary longer grains will keep their timbral identity.
Even if in the classical Granular Synthesis theory usually will be considered only symmetrical grain envelopes, a more flexible management of the parameters will result in a more musical content, where, for example, grains with slow attack and quick decay will generate a reverse-playback effect.
In addition to the typical linear and curvilinear envelopes, mathematician Gordon Monro of Sydney University advanced the employment of a complex envelope, created overlapping a sinusoidal envelope with any linear or exponential wave.

2.3 Granular Synthesis Algorithms

We can find four typical historical approaches to Granular Synthesis:

- Grids and Screens
- Pitch-Synchronous Granular Synthesis (PSGS)
- Quasi-Synchronous Granular Synthesis (QSGS)
- Asynchronous Granular Synthesis (AGS)

While the first three are synchronous algorithms, where grains follow others with the same time or

with linear functions, in the fourth one the grains order is programmed with probabilistic calculations.
Grids and Screens method was developed by Iannis Xenakis, who organized the grains sequences with screens that described grains frequencies and amplitudes at a certain time and with a certain density.
Pitch-Synchronous Granular Synthesis (PSGS) was theorized by De Poli and Piccialli to create pitched sounds with one or more formants in their spectrum, using FIR filters (finite impulse response) to synthesize grains on the basis of their spectrum analysis.
Quasi-Synchronous Granular Synthesis (QSGS) generates a flow of grains with variable delay time.
When delay time is the same for each grain, flow envelope can be approximated to a periodic function and QSGS can be analyzed as an AM Synthesis in which grain waveform is the carrier and grain envelope is the modulator.
With QSGS is also possible to recreate sound textures similar to Formant-Wave Synthesis (FOF), simulating human voice resonances or acoustic instruments. Moreover Barry Truax has experimented that working with irregular delay time the formant structures will distribute themselves around the carrier frequency, thickening the resulting sound.
With *Asynchronous Granular Synthesis* (AGS) the grains structure is managed in 'sound clouds', on the basis of probabilistic values for parameters like start time and duration of the cloud, duration of the grain, grain density, cloud frequencies width, cloud envelope and grains waveforms; the dynamic nature of the clouds allow to generate very complex and organic sounds.

2.4 The Betel Orionis system

Betel Orionis [7] is the module of SAIPH system dedicated to sound synthesis. It was developed by Giorgio Nottoli and Giovanni Costantini in 1998, and was born from the collaboration between University of Rome 'Tor Vergata' and Frosinone Conservatory.
Betel Orionis is based on the ORION DSP, designed by Giorgio Nottoli in 1991 to implement the main sound synthesis algorithms used in computer music.
ORION is composed by four arithmetic-logic units, three RAM and three I/O units and it is provided with micro instructions to generate audio primitives.
Executing music oriented digital signal processing algorithms, the computing work is distributed among the four ALUs to maximize parallelism while the data transfer work takes advantage of the three data rams structure.
The project of Betel Orionis is based on a multiprocessor architecture and implements all the most important algorithms for real-time sound synthesis with a capacity of eight stereo output audio channels. In particular, each channel is controlled by an Orion DSP driven in real-time by a host personal computer. Each Orion DSP can work alone or it can communicate with other Orions exchanging samples via a serial link. This feature allows the user to choose the best configuration for his own needs.
The whole system can be totally controlled through a three level architecture. At user interface level we can control multiple sound events together setting few and simple parameters in a gestural way (with computer keyboard, mouse, knobs and MIDI controllers). At control level we can take advantage of deterministic and stochastic methods implemented by a host computer, and at sound generation level we can use sound synthesis algorithms inside Orion DSP micro programs.

3 Texture

Texture is an instrument to perform Asynchronous Granular Synthesis based upon an algorithm developed by Giorgio Nottoli, and is fully implemented with programming language C++.
It generates a grains' flow in which the way each grain follows the other depends on probabilistic parameters such as grain density (attack/sec), overlapping, synchronism and fade of each grain.
The spectrum generated by the synthesis could be harmonic, expanded or contracted according to the value of frequency exponent parameter; the result will be a sound texture that can change from noisy fragmented sounds, to metallic and tuneless sounds such as bells, finally to harmonic sounds similar to strings or choirs.
Texture is available both as VST or Audio Unit instrument, and each interface's parameter is automated and can be controlled by the host; some factory presets are implemented in a combo box on theupper-left side of the interface, but user can save his own sound inside the host if needed.
An advanced and a basic interface are provided for expert users or casual ones; we're going to see the details in the following chapters.
The main panel, see Fig. 1, is on the upper-middle side of the interface and it contains a master volume, a sustain button to hold the root note, a graph button to hide/show the spectrum that we're going to generate, and a polyphony button to switch between monophonic and polyphonic mode.
In the polyphonic mode, we have a queue of 64 keys on which the partial harmonics of the spectrum will

be distributed; the queue will be cleaned when user stop pressing any midi note.
Frequency deviation knob affects the harmonic distribution of the spectrum around the frequency root (midi note pressed on the piano keyboard); it can change between 0 and 2500 cents of frequency.
The first partial and the number of partials knobs affect respectively the first partial and the number of partials in the generated spectrum and their values can change between 1 and 128.
Finally frequency exponent [or Steve McAdams exponent] knob can vary between 0 and 2, and it generates a harmonic spectrum when its value is 1, a contracted spectrum if the value < 1, or an expanded spectrum when the value > 1.
Waveform Panel defines the sound content of the grains.
User can set the number of oscillators used to generate the sound texture, choosing between 32, 63, 128 or 256 active oscillators.
Waveform knob will set the harmonic number of the precomputed waves; it can change between 1 (simple sine wave) and 32 harmonics. The amplitudes of each harmonic decrease on 1/n order, similar to the behavior of a saw wave.
The parameters in the Micropoly panel will affect the way each grain follow in time the other.
They are all probabilistic values, which mean each parameter sets the probability that the time event happens.
Density knob affects the number of attacks per second, and its value can change between 1 and 1000.
Overlapping knob sets on each instant the probability that the grain should continue to play, or it should start the release state of his envelope.
Finally the change knob affects the probability that the following grain should change his root frequency, so that in polyphonic mode it sets the rate each voice starts playing his sound.
Envelope panel affects the grain's envelope. Inside the panel user can see two joystick controller, the first one controls the attack and decay time of each grain (they can change between 1 and 1000 ms), and the second one sets the sync and fade parameters.
The sync parameter changes the probability that the grain starts playing together with the previous one, and the fade parameter affects the probability that a crossfade occurs between grains.
Then we have four sliders to control the ADSR of the note played; attack, decay and release are rates, and can change between 1/40 of second and 40 seconds, on the other hand sustain is an amplitude, and it can change between 0 and 1.

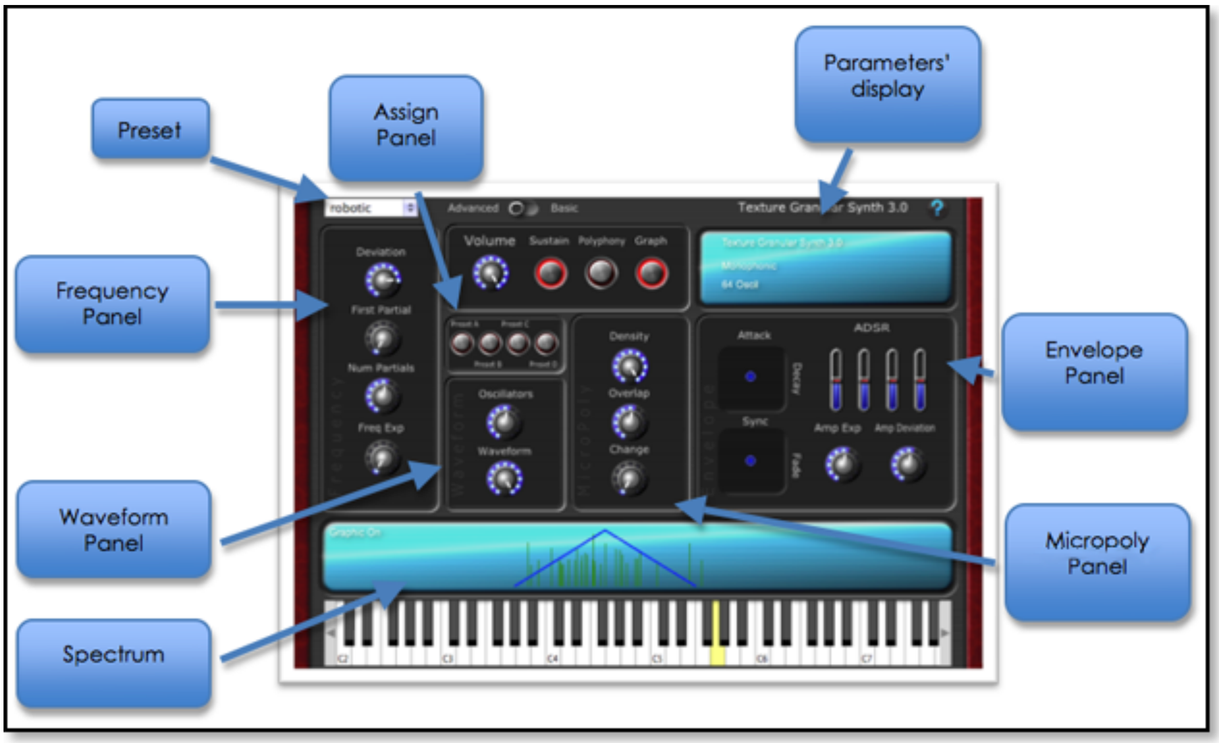


Figure 1: Main panel

Then user can control the amplitude exponent, which can change between 0 and 2, to modify the amplitude of the spectrum's partials; when the value is 0 the results is a plan spectrum (with the same amplitude for each partial), when the value = 1 the amplitudes will be decreasing with the order of 1/n, and when the value >1 the higher partials will fade.
Finally amplitude deviation randomizes the amplitude of each partial.
Assign panel contains four buttons in order to assign the current sound we're playing to the basic interface.
Each of the four preset corresponds to one of the four vertex of the pad in the basic interface.
The basic interface, see Fig. 2, gives casual and not expert users a tool to play sound textures generated by granular synthesis.
On the four vertex of the preset pad (the red rectangle) we have pre loaded four factory sounds, and user can move the joystick in every position inside the rectangle to mix the preset and create a unique combination of the four sounds.
User can also assign to each vertex a sound specifically created in the advanced interface, by pressing the four buttons in the assign panel. On the contrary user can create a sound texture by moving the joystick of the preset pad, then he can switch to the advanced interface to change the detailed parameters of the sound he has just chosen.

3.1 Technologies and development environment

Audio Units are audio plugins developed using Apple Core Audio technology, integrated into Mac OSX operating system. Audio Unit SDK is written in Objective-C language and its native development environment is XCode.
To develop Texture we have taken advantage of JUCE audio [14] libraries, that is a cross-platform toolkit written in C++ useful to create stand alone applications and audio plugins. It offers classes to manage MIDI, read and write audio files, draw waveforms, a Synthesizer class to create polyphonic synths and, mainly, wrappers to build VST (Windows, Mac OSX, Linux), AU (Mac OSX) and RTAS (Windows and Mac OSX) using the same source code. With JUCE is also possible to develop, with little modifications to the source code, mobile apps for iOS (iPhone & iPad).

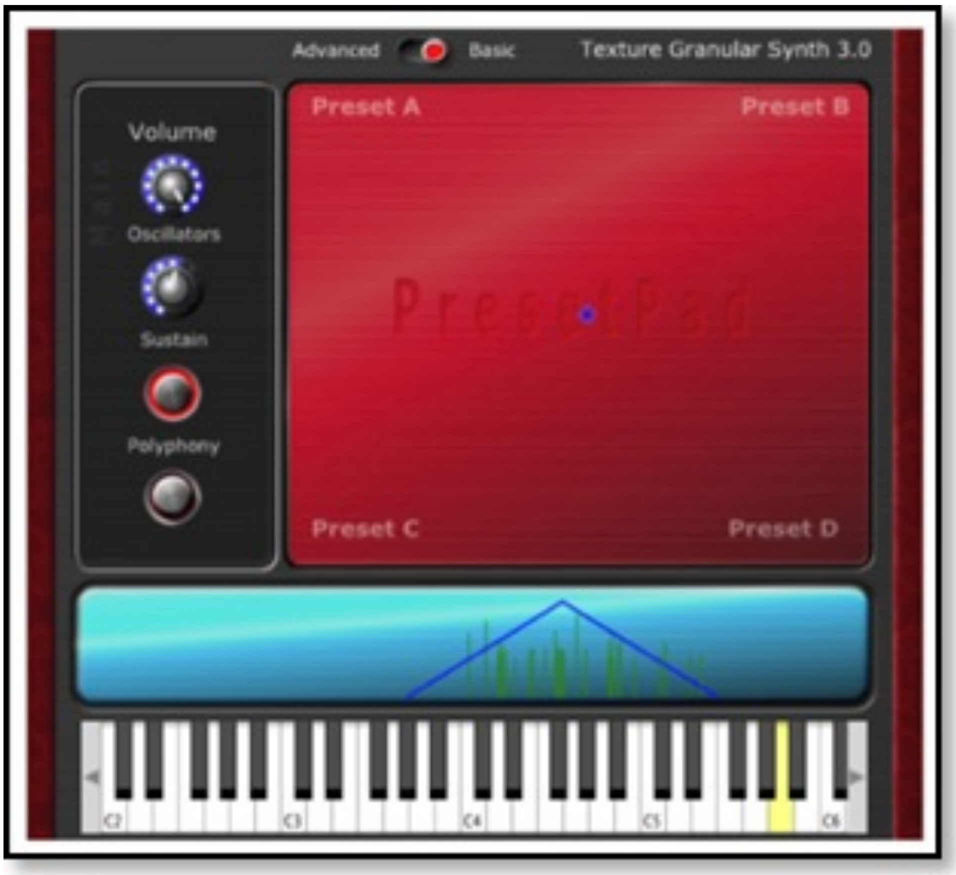


Figure 2: Basic interface

Finally, to build VST audio plugins we have to include into our project the Steinberg SDK, that will provide all the classes to manage the audio buffers and the plugin interface.

4 Real-time musical application

We tested our Texture Granular Synth by writing a real-time musical performance [15-19]. The musical instrument that we propose has been developed by using the Max/MSP [11] environment. It is constituted by three components: the control unit, the mapping unit and the Texture Granular Synth. The control unit allows the performer to control two parameters. Fig. 3 shows the Max/MSP patch that constitutes the control unit. Particularly, performer draws lines and curves in a bi-dimensional box, by clicking and moving mouse inside the patch itself. The control unit patch is constituted by the *lcd* Max/MSP object that returns x, y mouse space coordinates.

The evident points in the Fig. 3 represent the input/output patterns of the training set.

In detail, the two x, y control parameters don't influence directly the parameters that rule the behavior of the sound generators, but they are pre-processed by the mapping algorithm.

The implementation of musical expressivity is accomplished once we define the correspondence between the two x, y control parameters and the four synthesis parameters related to frequency panel (Fig. 1), that is to say, once we define the right mapping.

The chosen mapping strategies, by means of which the synthesis parameters are controlled, all influence the way the musician approaches the composition process [21-25].

In Fig. 4, the scheme of the virtual musical instrument is shown.

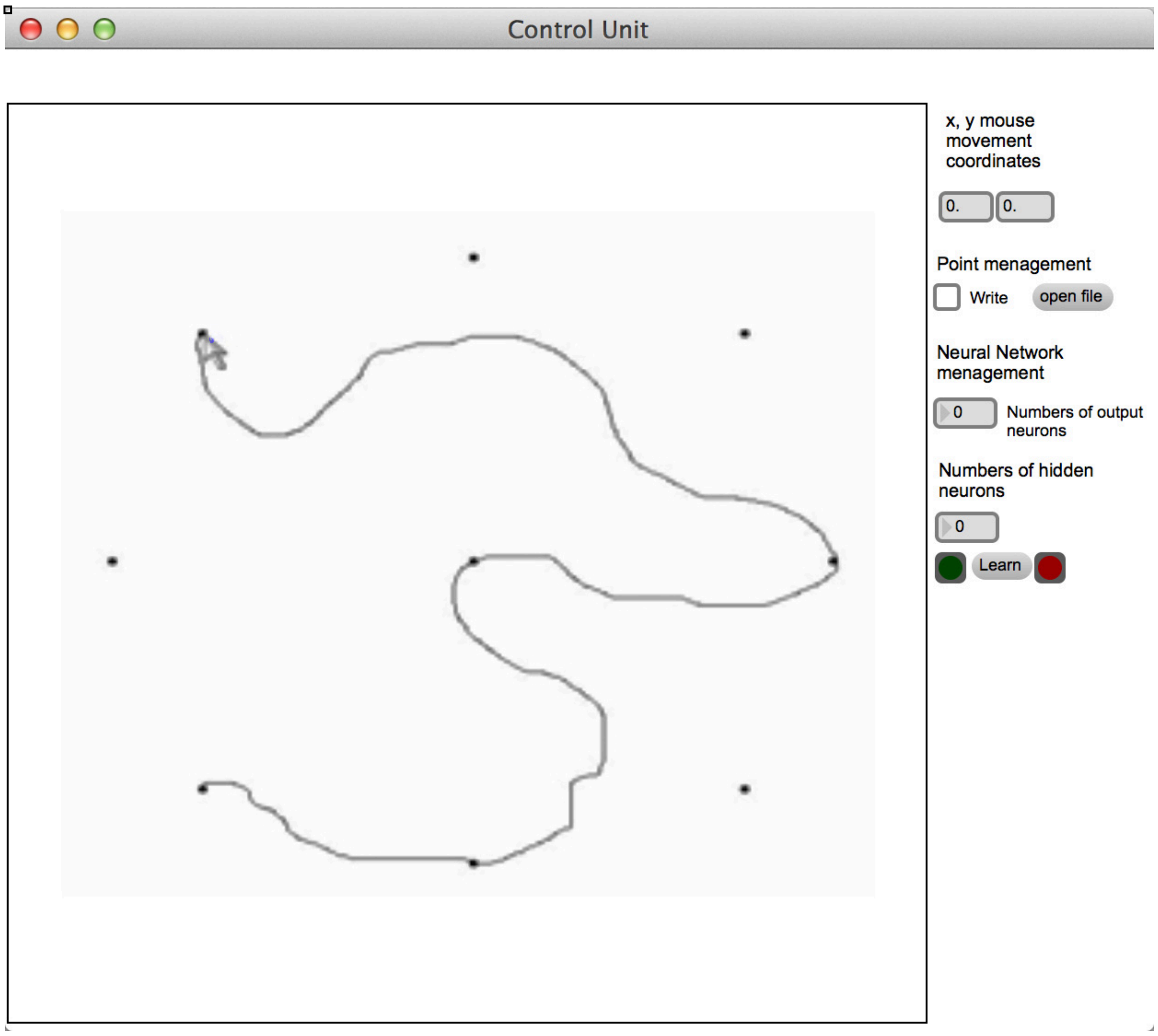
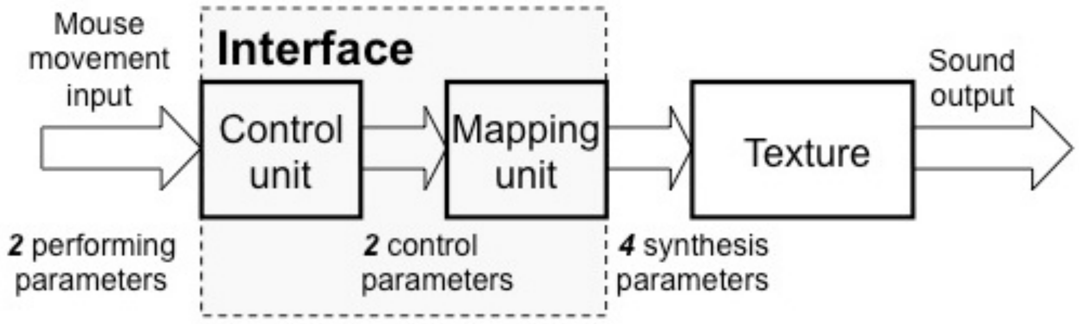

Control Unit
x, y mouse movement coordinates
0.
0.
Point menagement
Write
open file
Neural Network menagement
0
Numbers of output neurons
Numbers of hidden neurons
0
Learn

Figure 3: The control unit



Figur 4: The virtual musical instrument scheme

4.1 Musical expressivity implementation

To investigate the influence that mapping has on musical expression, let us consider some aspects of Information Theory and Perception Theory [25]:

- the quality of a message, in terms of the information it conveys, increases with its originality, that is with its unpredictability;
- information is not the same as the meaning it conveys: a maximum information message doesn't make sense, if any listener that's able to decode it doesn't exist.

A perceptual paradox [26] illustrating how an analytic model fails in predicting what we perceive from what our senses transduce is the following: both maximum predictability and maximum unpredictability imply minimum information or even no information at all.

A neural network approach [27] is chosen to exceed the perceptual limits above mentioned.

Let's assume the following concepts:

1. a predictable musical message be associated to an a priori known functional relation between the surface R^2 and the hyperspaces R^4, that is to say, between the set of all the 2 inputs and the set of all the 4 synthesis parameters;
2. an unpredictable musical message be associated to a non linear and a priori unknown correspondence between R^2 and R^4.

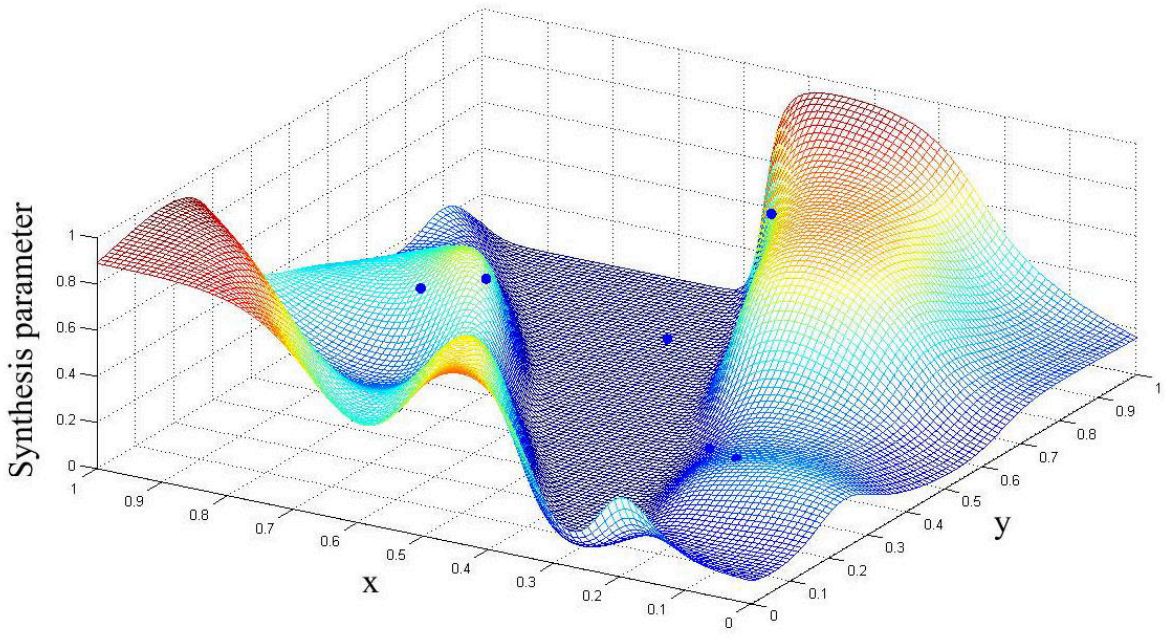


Figure 5: A synthesis parameter as a function of the input x and y.

A composer can easily follow the above assumptions by making use of a FFBPNN trained as follows:

1. he fixes a point in the 2-dimensional x, y space and he links it to a desired configuration of the m synthesis parameters;
2. he repeats D times step 1., so as to have D 2-to-4 examples at his disposal; they constitute the training set for the mapping unit;
3. he chooses the neural network structure [27], that is to say the number of hidden neurons to use; then, he trains the neural network.
4. he explores the 2-dimensional x, y input space by moving through known and unknown points, with the aim of composing his piece of music.

In Fig. 5 the slope of a synthesis parameter returned by the output of the neural network, as a function of the input x, y coordinates, is shown. The evident points in the graph represent the input/output patterns of our training set.

5 Conclusion

In this paper we presented a software module that performs real-time sound synthesis by means of the Granular Synthesis method. The software allows to control the frequency deviation, the number of partials, the Steve McAdams exponent, as well as the probabilistic distribution of the grains, and their values of attack, delay, sustain and release, allowing a huge variety of possible sound textures.

The system is designed to be considered as a virtual musical instrument for composing and performing expressive musical sound. We direct attention to common musical aesthetics as a determinant factor in musical expressivity.

The system is available as a VST or AU module, so it can be integrated with almost every audio software both for Window and for Mac. We also proposed some applications which run on Max/MSP: on the first one, the interface is formed by a control unit that supplies *x* and *y* coordinates of points in a bi-dimensional box, and by a mapping unit, based on a FFBPNN, that processes those points, in order to provide suitable relationships between input mouse movements and sound synthesis parameters. The second interface is a glove with bend sensors, which allows to control the system by means of hand movements.

The experiences made by working with our interfaces have shown that the mapping strategy is a key element in providing musical sounds with expressivity.

References:

[1] Roads C., The computer music tutorial, The MIT Press, 1996.

[2] Bosi M., Goldberg R. E., Introduction to Digital Audio Coding and Standards (The Springer International Series in Engineering and Computer Science), Springer, 2003.

[3] Fletcher N., H., Rossing T., D., The Physics of Musical Instruments, Springer, 2005.

[4] S. Cavaliere, G. Di Giugno, E. Guarino, "MARS: The X20 device and SM1000", Proceedings of the ICMC, pp. 348-351, S. Jose, 1992.

[5] P. Andrenacci, E. Favreau, N. Larosa, A. Prestigiacomo, S. Sapir, "MARS: RT20M/EDIT20 - Development tools and graphical user interface for sound generation board", Proceedings of the ICMC, pp. 340-343, S. Jose, 1992.

[6] Kyma workstation, documentation available at http://www.symbolicsound.com/cgi-bin/bin/view/Products/WebHome

[7] Nottoli G., Salerno M., Costantini G., A new interactive performance system for real-time sound synthesis. In: Proc. of International Computer Music Conference '98. Ann Abor, Michigan, USA, October 1-6, 1998, p. 33-36.

[8] Nottoli G., Salerno M., Costantini G., Sabatini A., A multiprocessing system for real-time sound processing and spatialization. In: Proc. of EURASIP Conference for Multimedia Communications and Services '99. June 24-26, 1999, Kraków, Poland.

[9] Steinberg VST Audio Plug-Ins SDK, 3rd party developer support site, available at http://www.steinberg.net/324_1.html

[10] Cycling74 Max/MSP documentation, available at http://www.cycling74.com/products/maxmsp

[11] Texture Granular Synth 3.1 download, at http://www.mastersonicarts.uniroma2.it/t_research/texture.html

[12] Roads C., Travis Pope S., Piccialli A., De Poli G., Musical Signal Processing (Studies on New Music Research, 2), Routledge, 1997.

[13] Xenakis I., Formalized Music, thought and mathematics in composition, Harmonologia series no. 6, Pendragon press Stuyvesant NY, 1965.

[14] JUSE, documentation available on the web at http://www.juce.com/about-juce

[15] Bongers, B. 2000, Physical Interfaces in the Electronic Arts. Interaction Theory and Interfacing Techniques for Real-time Performance, In M. Wanderley and M. Battier, eds. Trends in Gestural Control of Music. Ircam - Centre Pompidou.

[16] Costantini G., Todisco M., Carota M., Maccioni G., Giansanti D., A New Adaptive Sensor Interface for Composing and Performing Music in Real Time. In: Proc. of IWASI '07 (IEEE International Workshop on Advances in Sensors and Interfaces). Bari, Italy, June 26-27, 2007, p. 106-110

[17] Costantini G., Todisco M., Carota M., Casali D., A new physical sensor based on neural network for musical expressivity. In: Proc. of 13th Italian Conference Sensors and Microsystems. Rome, Italy, February 19-21, 2008, p. 281-288

[18] Orio, N. 1999, A Model for Human-Computer Interaction Based on the Recognition of Musical Gestures, Proceedings of the 1999 IEEE International Conference on Systems, Man and Cybernetics, pp. 333-338.

[19] Costantini G., Todisco M., Saggio G., A Cybernetic Glove to Control a Real Time Granular Sound Synthesis Process. In: Proc. of IMCIC International Multi-Conference on Complexity, Informatics and Cybernetics. Orlando, Florida, USA, April 6-9, 2010

[20] Odowichuk G., Trail S., Driessen P., Nie W., Page W., Sensor fusion: Towards a fully expressive 3D music control interface. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PacRim), 2011.

[21] Costantini G., Todisco M., Saggio G., A Wireless Glove to Perform Music in Real Time. In: Proc. of WSEAS International Conference on Applied Electromagnetics, Wireless and Optical Communications. Penang, Malaysia, March 23-25, 2010, p. 49-54

[22] Aramaki M., Kronland-Martinet R., Ystad S., Perceptual Control of Environmental Sound Synthesis Speech, Sound and Music Processing: Embracing Research in India Lecture Notes in Computer Science Volume 7172, 2012.

[23] Costantini G., Saggio G., Todisco M., A Glove Based Adaptive Sensor Interface for Live Musical Performances. In: SENSORDEVICES International Conference on Sensor Device Technologies and Applications

[24] Saggio G., Giannini F., Todisco M., Costantini G., A Data Glove Based Sensor Interface to Expressively Control Musical Processes. In: 4th IWASI IEEE International Workshop on Advances in Sensors and Interfaces.

[25] Abraham Moles, Information Theory and Aesthetic Perception, University Of Illinois Press (1969).

[26] Rudolf Arnheim, Entropy and Art: An Essay on Disorder and Order, University of California Press (January 29, 1974).

[27] Hertz J., A. Krogh & R.G. Palmer, Introduction to the theory of neural computation, Addison-Wesley Publishing Company, Reading Massachusetts, 1991.