# Granular Synthesis with Cmix and MAX

Mara Helmuth

Columbia University
Music Department
New York, NY 10027

Texas A & M University
Music Department
College Station, TX 77843

mara@woof.music.columbia.edu, mara@silvertone.princeton.edu

**Abstract:**

Several granular synthesis programs on the NeXT will be shown. *StochGran* is an Interface Builder Objective C interface to several Cmix instruments. In this new version, sampling granular synthesis, more graphical features, and in-phase correlation of grains have been added. The second part of the demonstration features a granular sampling MAX patches running in real time on the IRCAM Signal Processing Workstation. Excerpts from a composition by the author for eight instruments and ISPW, *Evolutions,* which was created with both *StochGran* and Max, will be discussed.

## 1 Introduction

Construction of complex sound from grains, or small units of sound is known as granular synthesis. Probability equations are often used to control the large amount of parameter data for the grains. This technique was first described by Iannis Xenakis [1971] in his discussion of stochastic music. Barry Truax [1988] and Curtis Roads [1988] have both explored these techniques. *StochGran* is a NeXTstep interface to a Cmix implementation of granular synthesis instruments. The current version of *StochGran* incorporates new graphical features for more spontaneous control over sound creation, and the granulation of sampled sound. Real time granular synthesis is possible with the IRCAM workstation. The MAX graphical programming language has been used to create granular sampling patches for composition and stochastic improvisation. The computer part of *Evolutions*, a piece for eight live instruments and computer is created by MAX patches both playing Cmix-created sound files, and performing granular sampling in real time on sampled acoustic instrument sounds.

## 2 Cmix under NeXTstep Implementation: *StochGran*

*StochGran* allowed the composer to control a large number of grain parameters stochastically and independently. In the original version the parameters were typed in manually for each of the thirty-six fields. A Cmix score or data file was written out, the *sgran* stochastic granular synthesis instrument was run with these parameters, and a sound file was written to the disk. The sound file could be played

within the program. The details are described in Helmuth [1991]. Because the program was non-real time, an unlimited number of grains could be layered, the grains could be any duration, and control functions could be as complex as necessary. Several features have been added to aid composition, including sampling sound files and graphical editing of control functions.

### 2.1 Granular Sampling

*StochGran* has been modified to read sound files, as well as synthesizing from waveforms. Using sampled sound as the basic unit of construction has been termed granular sampling [Lippe, 1992]. The amount of time to skip into the file, and rate at which to proceed through the file while writing out processed grains was specifiable. As in the synthesis version, the probabilities of various pitches, rates and durations to occur at each point in the event were all controllable. In this case, grain frequency was a result of a transposition. Because the phase of the grain affects how the timbre of the resulting sound will be perceived (adjacent grains with widely varying phases can sometimes produce a noisy sound), control over phase was important. An algorithm to line up the phases of overlapping grains described by Jones et al., [1988] has been implemented in the new version.

### 2.2 Graphical Function Editing

Control functions determine the shape of change of grain parameters over the course of an event. In Cmix, one types the function parameters into a line of data. Previously *StochGran* allowed one to select linear or exponential shapes. To create complex functions one had to return to line editing. All four control functions have been made editable graphically. This feature was easily implemented due to Fernando Lezcano's program, *EnvelopeEd,* which provided the EnvelopeView graphical object. While the previous version of *StochGran* was a vast improvement over tedious data file editing, with the graphical function editor the composer has been able to think more clearly and specifically about the shape of parameter changes in phrases and musical gestures.

## 3 MAX/ISPW Implementation

MAX patches have been constructed to do granular sampling on the IRCAM Workstation (ISPW) in real time. The music programming language MAX allows the user to connect signal processing object graphically, [Puckette, et al., 1990]. The goal was to create the same types of sounds as with *StochGran*, but in real time. This would allow improvisation, and facilitate performance with instrumentalists. While I had attempted improvisation with my *PieceNow* interface, alternately digitizing live acoustic performers and then processing the sounds and playing them, and even before that with a set of shell scripts, the results depended more on the live component than the labored computer part. To improvise with granular synthesis sounds required the ISPW, and a set of controls with which one could manipulate a number of parameters simultaneously. The possibility of having instrumental and computer parts interact freely in time was attractive. Several important issues came up in the translation of the Cmix programs to MAX. It was necessary to change the method of grain calculation to create the stochastic sounds in real time. The compositional process was somewhat different, and choices about how to trigger events in performance had to be made.

### 3.1 Description of the Patches

Space does not allow a detailed description of the Max patches. The algorithm is similar to the one described for the Cmix *StochGran sgran* instrument, with exceptions made for the real time aspects discussed below. A simple probability table representing a quasi-normal distribution was used instead of the equation in *sgran*. Grain parameters were sent in either with beginning and ending event settings to move between over a particular time, or ending settings to move toward. The settings were

sent either from specific buttons designed for a certain sound, or manipulated individually with sliders and sent when desired in the *testparams* patch. The parameters were received by a patch which applied the event amplitude envelope and the voice volume. This patch in turn sent the data to a subpatch which calculated the incremental change in parameters throughout the event. This patch sent the information to the lowest level patch, which read the sample table at the appropriate rate, duration and transposition, and applied the grain envelope. The highest level patch had the user controls for reading sound files from the disk, master volume, reverb amplitude and time, and buttons to initiate events in various sections of a piece.

### 3.2 Differences in Musical Strategy using a Real-Time Platform

The most obvious difference between real time and non-real time systems was in performance. A strategy was needed for making the computer events happen, either under human or program control, or some combination of the two. How were the computer and live performers to interact? The design of this aspect had profound consequences for a composition.

Another important difference is the limitation on processor power. While the ISPW is extremely powerful, layering thousands of simultaneous grains with several different types of room simulation, as I have done in Cmix, was beyond even the capabilities of the ISPW. To make efficient use of resources, different patches were set up for different situations which emphasize the techniques currently needed, rather than relying upon one or two multipurpose programs. Phase correlation was not used; a low pass filter removed noise. Four to eight or more "voices" were created, and each calculated grains linearly, without overlaps, rather than allowing stochastically-made decisions to generate grains in a more random order. The start times were delayed stochastically, and the musical effects of the sounds were similar to many of the Cmix-created files. Metronomes for each voice triggered the sending of new grain parameters on the millisecond level, so that complex changing arrangements of grains were possible. Where extremely dense layerings were needed, sound files were written to the disk and read back.

A third contrast in working with a real time system was the compositional process. When composing with Cmix (after the instrument was created), a pattern of waiting for the sound to be created, playing, and editing the data was established. Because it took from a few seconds to hours for a dense mass of grains with long durations, a thoughtful approach was encouraged. There was a definite joy to being able to design the processing effect of a button on a sound, and test it out immediately in MAX. With MAX, however, the delay was in the programming, not the realization of sound. Complex programs were more time consuming to write in MAX than in C, as the handling of strings and messages was difficult.

## 4 A Compositional Example: *Evolutions*

*Evolutions*, for flute, clarinet, alto saxophone, two percussion, ISPW, harp, violin and cello, was written to make use of a wide variety of the granular synthesis sounds, using instrumental sounds as the source, to produce timbral evolutions and gestures.

### 4.1 Timbral Evolution

Granular synthesis allows one precise control over a very wide range of timbres. In this composition the source sounds were sampled instrumental sounds. Sound files were read onto the hard disk and read into tables as needed. Sampling the instruments live would have required more rehearsal time than was feasible. There were about two hundred three-second files, including a hundred percussion sounds. Each instrument was sampled at pitch intervals of a second or third, and some were sampled at different tone colors such as violin played sul ponticello, or clarinet played very softly. Any of these files could be read into

any of the voice tables stored in memory on the processor, at any time. Since events of changing parameters are created independently on any of the voices, the timbral and textural variety was large, and the ability to move from any tone color to another was unlimited, at any rate or shape of change. Parameters such as pitch depended on the interaction of several parameters, sound file frequency and grain rate, which allowed for interesting variations as the rate and frequency could change independently.

### 4.2 Musical Gesture

With the changing parameters one can specify contrasting and similar movements throughout an event. An internal timbral counterpoint can be described as a musical gesture. This term is appropriate because it refers to an unity of conception of the sound. A physical gesture made by a living organism may seem one connected movement, although it involves the individual movements of many muscles. Similarly, the timbral gesture may be composed of many gradual parameter changes in grain rate, duration and transposition, but still heard as one unique musical unit.

## 5 Summary

Two different implementations of granular synthesis techniques on the NeXT have been constructed. *StochGran*, a NeXT interface to a Cmix instrument, is available for anonymous ftp at the Princeton.edu archive site. The MAX patches performed granular sampling in real time using the IRCAM Signal Processing Workstation. The composition *Evolutions* illustrated the use of both techniques in a composition for instruments and real time signal processing. The MAX patches are available if a request is sent to the email address above.

## References


[Helmuth, 1991] Helmuth, Mara. Patchmix And StochGran. *Proceedings of the International Computer Music Conference.* Montreal: McGill University. 1991.

[Jones, et al., 1988] Jones, D. and Parks, T. Generation and Combination of Grains for Music Synthesis. *Computer Music Journal* 12(2):27 - 34, 1988.

[Lansky, 1990] Lansky, Paul. The Architecture and Musical Logic of Cmix. *Proceedings of the International Computer Music Conference.* Glasgow: ICMC Glasgow 1990, 1990.

[Lippe, 1993] Lippe, Cort. A Musical Application Using the IRCAM Signal Processing Workstation. Talk given at the *1993 Science and Music Conference.* London.

[Puckette, 1988] Puckette, Miller. Combining Event and Signal Processing in the MAX Graphical Programming Environment. Computer Music Journal 15(3):68 - 77, 1988. Cambridge, MA: Massachusetts Institute of Technology.

[Roads, 1988] Roads, Curtis. Introduction to Granular Synthesis. *Computer Music Journal*, 12(2), 1988. Cambridge, MA: Massachusetts Institute of Technology.

[Truax, 1988] Truax, Barry. Real-Time Granular Synthesis with a Digital Signal Processor. *Computer Music Journal*, 12(2), 1988. Cambridge, MA: Massachusetts Institute of Technology.

[Xenakis, 1971] Xenakis, Iannis. 1971. Formalized Music. Bloomington, IN: Indiana University Press.